Bachelor-Thesis

OPEN SOURCE FÜR DAS SMART HOME

vorgelegt von:

Lukas Müller

im Studiengang Medien und Informationswesen

Sommersemester 2017

Erstbetreuer:

Prof. Dr. rer. nat. Tom Rüdebusch

Zweitbetreuer:

Prof. Dr. rer. pol. Volker Sänger

HOCHSCHULE OFFENBURG

FAKULTÄT MEDIEN UND INFORMATIONSWESEN

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Während Menschen schon seit langer Zeit angefangen haben ihre Häuser zu automatisieren (Bewegungsmelder, Rollladensysteme, …), gewinnt erst seit kurzer Zeit der Begriff „Smart Home“ an Popularität. Nicht nur im Kontext der Industrie, wo ganze Gebäudekomplexe durchautomatisiert werden, sondern auch für den durchschnittlichen Verbraucher, der mehr Intelligenz in sein Privathaus bringen möchte, wird dieser Technologiezweig immer attraktiver. Während auf dem Markt schon viele verschiedene Technologien existieren, ist die Vernetzung der einzelnen Geräte eine Aufgabe, die von einer zentralen Plattform übernommen werden muss. Solche Plattformen existieren zweierlei: Proprietär, sprich der Code ist nicht frei verfügbar und ggf. muss der Benutzer den Service bezahlen, und Open Source, also das genaue Gegenteil, was Gegenstand dieser Arbeit sein soll.

Im Rahmen dieser Bachelor Thesis soll zunächst ein Überblick über die involvierte Technik und die verschiedenen Plattformen geliefert werden um dann im Anschluss anhand eines beispielhaften Smart Home Szenarios, welches repräsentativ für eine private Hausautomatisierung sein soll und gleichzeitig populäre Technologien verwenden soll, die Plattformen miteinander zu vergleichen. Der Vergleich findet anhand von vorher bestimmten Kriterien statt, die die Plattformen auf Usability, Kompatibilität und Support prüfen.

Die Arbeit lässt sich grob in drei Kapitel aufteilen, wobei Kapitel 1 sich mit den grundlegenden Konzepten befasst. Im ersten Unterkapitel (1.1) wird der Begriff „Open Source“ beleuchtet, im darauffolgenden Kapitel (1.2) das Konzept von Smart Home, inklusive seiner Definition, Geschichte und grundlegenden konzeptionellen Informationen. Außerdem werden wichtige Technologien vorgestellt, die im Kontext von Smart Home relevant sind, wie der Hardware (1.2.4), den verwendeten Protokollen (1.2.5) und den Plattformen selbst (1.2.6), im letzten Kapitel wird außerdem anhand der Verbreitung entschieden, welche Plattformen Teil des Vergleichs werden.

Das zweite Kapitel beinhaltet die Umsetzung des Beispielszenarios in den verschiedenen Plattformen. Hier wird zunächst das Szenario selbst und die Geräte vorgestellt, die das Szenario verwendet (2.1). Anschließend folgt die Umsetzung selbst (2.2), inklusive eines Kapitels, in dem auf diejenigen Werkzeuge eingegangen wird, die in allen Umsetzungen verwendet werden (2.2.1). Das Kapitel danach beinhaltet den eigentlichen Vergleich anhand der bereits vorgestellten Kriterien (2.3).

Im letzten Kapitel (3) wird ein Fazit gezogen und gleichzeitig ein Ausblick für die Zukunft des Marktes versucht.

1 Open Source und Smart Home

1.1 Open Source

Der Term “Open Source” beschreibt eine Art der Technologieentwicklung, bei der die zugrundeliegenden Quellcodes, Baupläne oder jegliche andere Dokumentation der Öffentlichkeit frei zugänglich ist. Entgegen der verbreiteten Erwartung, deckt dieses Konzept nicht nur die Entwicklung von Software ab, sondern auch anderer Technologiezweige, wie Hardware, Robotertechnik oder Pharmazie. Die vorliegende Arbeit konzentriert sich jedoch auf die Verwendung von Open Source Software.

Im Gegensatz zu Proprietärer Software, welche die Nutzung, Verbreitung und Veränderung jener stark einschränkt, hat die Community einer Open Source Software die Möglichkeit den Source Code, also essentiell die Funktionsweise der Technologie, frei einzusehen und ist somit in der Lage und auch rechtlich befugt sie auf jede Art und Weise zu verwenden. Dies beinhaltet unter anderem auch die Veränderung und damit die Weiterentwicklung der Software. Hintergrund des Konzepts ist der Gedanke des Programmierers Richard Stallmans, dass wissenschaftliche Erkenntnisse - so auch die Funktionsweise von Computer Programmen - geteilt und verbreitet werden müssen. [1]

1.2 Smart Home aus Entwicklersicht

1.2.1 Definition

Schon seit langer Zeit existieren Konzepte im Thema Hausbau, die mit dem Begriff „Smart Home“ verbunden werden könnten: effiziente Platznutzung, hochentwickelte Bautechniken, die Nutzung von Solarpanels oder auch moderne Wasseraufbereitungssysteme. Das alles fällt jedoch nicht unter die Definition des Begriffs. Diese lautet im Oxford Dictionary folgendermaßen: [2]

> *A home equipped with lighting, heating, and electronic devices that can be controlled remotely by smartphone or computer.*

Essentiell sind also (fern-)steuerbare, vernetzte Geräte, die auch zu Automationen zusammengeschlossen werden können. Ziel des Verfahrens ist die Erhöhung der Lebensqualität und Sicherheit der Bewohner und außerdem eine effizientere

Energieverwertung. Smart Home ist als Konzept nicht nur auf Privathäuser anwendbar, sondern findet auch Gebrauch in industriellen Gebäudesystemen, dort jedoch schon weitaus länger. Begriffe wie Hausautomation, Intelligentes Wohnen, Gebäudeautomation oder eHome, sind weitestgehend gleichbedeutend mit Smart Home. [3]

1.2.2 Geschichte

Es fingen zwar schon Ende der 1960er Jahre einige Hobbybastler an, ihre Häuser auf eine Art und Weise zu modifizieren, die man „smart" nennen könnte, der eigentliche Begriff kam jedoch erst später zum Einsatz. Es gelten allgemein zwei Ereignisse als die Hauptkatalysatoren des Smart Home Konzepts: Um 1915 herum begannen elektronische Haushaltsmaschinen Einzug in bürgerlichen Häusern zu nehmen. Geräte wie Staubsauger, Küchenmaschinen oder Nähmaschinen wurden als zeitsparend beworben, obwohl Elektrizität noch nicht sehr verbreitet war. Es dauerte bis in die 1950er Jahre bis der Großteil der Haushalte mehr als nur Licht und eine einzige Steckdose hatten. Mit dem Umschwung des Ideals der Hausfrau steigerte sich das Interesse nach arbeitssparenden Geräten noch ein Mal. Doch erst nach dem zweiten großen Ereignis, dem Einzug des Personal Computers und mit ihm vielen anderen Multimedia Geräten, begann die Smart Home Technologie überhaupt Sinn zu ergeben.

Kommerzielles Interesse an Hausautomatisierung führte zur Gründung der Interessenvertretung „Smart House" von der National Association of Home Builders der USA, welche als Ziel hatte die notwendige Technologie in den modernen Hausbau einzubringen. Das Konzept erlangte eine gewisse Prominenz, nachdem es in großen Magazinen wie Vanity Fair behandelt wurde. Nichtsdestotrotz ist die tatsächliche Verwendung in privaten Haushalten noch eher gering, wobei sie stetig wächst. [4, 5]

Mit der Weiterentwicklung von Technologien wie Sensoren, Prozessoren und Kommunikationssystemen, wurde die Entwicklung von Smart Home Szenarien mit der Zeit immer einfacher umzusetzen und zu handhaben. Darüber hinaus hat auch ein Abfall des Preises von verschiedenen Hardware Komponenten dazu geführt, dass es auch für weniger wohlhabende Haushalte nicht ausgeschlossen ist, solche Technologien zu verwenden. [6]

1.2.3 Grundlagen/Architektur

Grundsätzlich besteht ein Smart Home System aus drei wichtigen Komponenten: [6]

1) Physikalische Komponenten – Sensoren, Aktoren, Microcontroller
2) Kommunikationssystem – Kabel, Kabellos
3) System zur Informationsverarbeitung und Steuerung

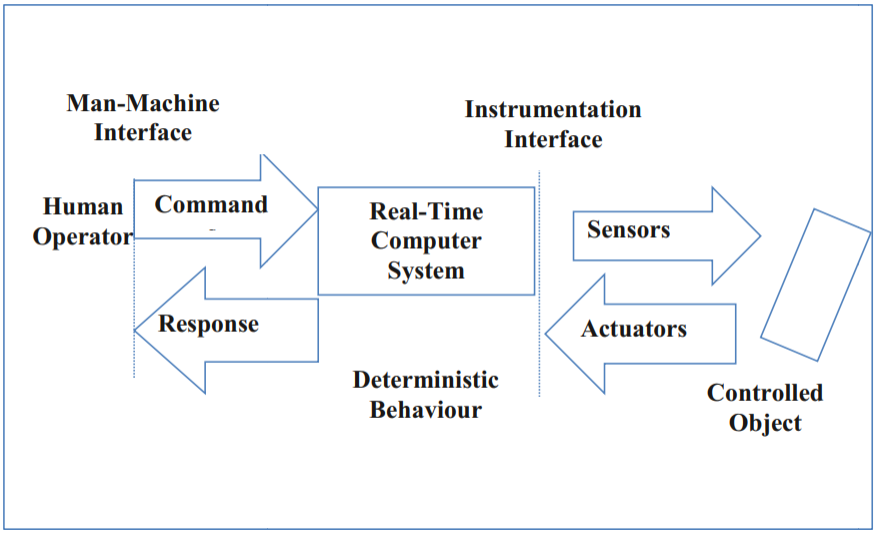

Abbildung 1: Zusammenspiel der Komponenten eines Smart Home Systems [6]

Die physikalischen Komponenten stellen die Interaktion mit dem Haus und der Umgebung dar, sie nehmen Signale wahr und führen die automatisierten Prozesse aus, mit denen jene Signale assoziiert sind. Das Kommunikationssystem stellt in den meisten Anwendungsfällen ein Drahtloses Funksystem dar oder auch im klassischen Anwendungsfall, eine Verkabelung. Unabhängig davon, wie es umgesetzt wird, dient es dazu, die Signale von den Sensoren an das Steuerungssystem und von jenem wiederum die Anweisungen an die Aktoren weiterzuleiten. Das Steuerungssystem selbst verarbeitet alle Informationen, die von Sensoren oder Aktoren an es gesendet werden. Eine wichtige Unterscheidung muss zwischen dem Steuerungssystem und der physikalischen Komponente, auf der eben jenes betrieben wird, gemacht werden. Beim Steuerungssystem handelt es sich um eine Software, die dafür sorgt, dass Komponenten im Smart Home Netzwerk kommunizieren können und deren Information verarbeitet. Diese laufen meist auf Microcontrollern, welche aber zu den physikalischen Komponenten gezählt werden müssen. [6]

Zwar basiert auch ein großer Teil von Smart Home auf Automatisierungen, die nur ausgelöst von der Umwelt stattfinden, wie Temperaturschwankungen, Sonnenstand oder Regen, der Mensch spielt mit seinen Aktionen jedoch immer noch eine große Rolle. So können Automatisierungen mit dem Druck einer Fernbedienungstaste ausgelöst werden oder auch mit dem Betätigen des Lichtschalters. Außerdem muss nicht zwangsläufig alles über Automationen ablaufen, die direkte Steuerung von Aktoren wird durch das Steuerungssystem auch gewährleistet.

1.2.4 Hardware

1.2.4.1 Sensoren

Bei der Heimautomatisierung geht es im Endeffekt darum, Prozesse auszuführen, wenn bestimmte Ereignisse geschehen. Wie auch ein Mensch auf seine Umwelt reagiert, soll das System auf die Geschehnisse in seiner unmittelbaren Umgebung reagieren und dementsprechende Aktionen ausführen. Um bei der Analogie zu bleiben: Der Mensch nimmt seine Umgebung mit seinen Sinnen wahr, ohne diese wäre sein hochentwickeltes Hirn wenig hilfreich, da es keinen „Input“ hätte. Ebenso muss das System in der Lage sein, visuelle, auditive und andere physische Vorkommnisse wahrzunehmen, da es sonst wenig Sinn hätte, einen komplexen Computer zu betreiben, wenn dieser keine Möglichkeit hätte, auf seine Umgebung zu reagieren. [7] Hierzu dienen die Sensoren, welche die erste Hälfte der physikalischen Komponenten darstellen. Sie können grob in vier Kategorien aufgeteilt werden: [6]

Art der Wahrnehmung	Wahrnehmungsparameter	Anwendungsgebiet
Umgebung	Druck, Temperatur, Feuchtigkeit, Lichtstärke	Gesundheits- und Sicherheitssysteme, Energieeffizienz
Bewegung, Anwesenheit	Position, Drehgeschwindigkeit, Beschleunigung, Richtung	Sicherheit, Ortserkennung
Biochemische Stoffe	Feste Stoffe, Flüssigkeiten, Gase	Sicherheit, Gesundheitsüberwachung

Multimedia	Ton und Bild	Objekterkennung, Kontrolle, Spracherkennung, Kontextverstehen

Tabelle 1: Arten der Wahrnehmung, Wahrnehmungsparameter und Anwendungsgebiete

Im Folgenden werden die relevantesten Arten von Sensoren beschrieben und prominente Produktbeispiele vorgestellt. Mehr praktische Beispiele folgen im Kapitel „Ein exemplarisches Smart Home Szenario“, wo die für das Projekt verwendeten Produkte beschrieben werden.

1.2.4.1.1 Bewegungsmelder

Bewegungsmelder, nicht zu verwechseln mit Bewegungssensoren, sind elektronische Sensoren, die Bewegungen in ihrer Umgebung mit Hilfe von optischen Impulsen feststellen und diesen Impuls weiterleiten. Sie werden meist zum Einschalten einer Beleuchtung oder zum Auslösen eines Alarms verwendet. Bewegungsmelder können entweder aktiv mit elektromagnetischen Wellen, mit Ultraschall oder passiv anhand der Infrarotstrahlung der Umgebung arbeiten. Letzteres sind besonders mit den PIR-Sensoren (*Passive Infrared Sensor*) die verbreitetsten Geräte im heutigen Gebrauch. Sie stellen Temperaturveränderungen fest, beispielsweise wenn eine Person am Sensor vorbeiläuft. [8]

Bewegungsmelder sind ein wiederkehrender Bestandteil von Hausautomatisierungen, da sie keine „aktive“ Beteiligung des Nutzers voraussetzen, wenn sie richtig installiert sind. So würde man beispielsweise einen Bewegungsmelder in einem Hauseingang anbringen, wo der Hausbesitzer des Öfteren vorbei läuft wodurch wiederum eine Automatisierung ausgelöst werden kann. Im Gegensatz zu anderen Sensoren, waren Bewegungsmelder aufgrund ihrer großen Nützlichkeit auch schon Teil der Hausautomatisierung bevor der Begriff „Smart Home“ sich etablierte, als sie in einfachen Schaltungen mit Lampen verbunden wurden.

1.2.4.1.2 Temperatursensoren

Hierbei handelt es sich um einen Sensor, der die selbe Rolle wie ein Thermometer übernimmt, also die Temperatur messen und wiedergeben, meist in Form eines elektrischen Signales. Temperatursensoren haben sehr viele unterschiedliche Funktionsweisen. So gibt es beispielsweise Bauteile, die basierend auf der Temperatur ihren Widerstand ändern (Heißleiter oder Kaltleiter), Bauteile, die direkt ein verarbeitbares elektrisches Signal liefern, wie Halbleitersensoren oder verschiedene andere Verfahren, die zum Beispiel eine Resonanzfrequenz liefern. [9]

Temperatursensoren finden viel Einsatz in Hausautomatisierungen. So können zum Beispiel Fenster oder Rollladen geöffnet bzw. geschlossen werden oder Markisen ein bzw. ausgefahren werden, wenn es zu heiß oder zu kalt in einem Raum wird. Die Sensoren können auch im Außenbereich installiert werden, wodurch man eine genauere Kenntnis über die Außentemperatur hat, als wenn man einen Wetterdienst verwendet.

1.2.4.1.3 Luftfeuchtigkeitssensoren

Sensoren, die die Luftfeuchtigkeit messen, werden allgemein auch Hygrometer genannt. Die Verfahren, die hierfür verwendet werden, sind unzählig, meist verändert sich ein Material, wenn es größerer Feuchtigkeit ausgesetzt wird und anhand dieser Veränderung wird dann die relative oder absolute Luftfeuchtigkeit berechnet. Moderne elektronische Absorbtionshygrometer basieren dagegen jedoch auf der Veränderung der elektrischen Eigenschaften eines Sensors. So existieren beispielsweise kapazitative Sensoren, die bei zunehmender Feuchtigkeit ihre elektrische Kapazität verändern, und Impendanzsensoren, deren ohmscher Widerstand sich je nach Feuchtigkeit verändert. Darüber hinaus gibt es chemische, coulometrische oder auch optische Verfahren, um die Luftfeuchtigkeit zu bestimmen. [10]

Luftfeuchtigkeitssensoren können in der Hausautomatisierung in ähnlichen Anwendungsfällen wie auch Temperatursensoren verwendet werden, um beispielsweise ein Badezimmerfenster zu öffnen, wenn die Luft besonders voll mit Wasserdampf ist, nach einer Dusche also. Ein anderer Use Case wäre das Einschalten einer Dunstabzugshaube.

1.2.4.1.4 Luftqualität bzw. VOC Sensoren

Ähnlich wie Luftfeuchtigkeitssensoren, machen diese Geräte Aussagen über die Beschaffenheit der Luft innerhalb Häuser. Da die Luftqualität in Häusern durchschnittlich 8 Mal schlechter ist, als die der Luft draußen und in bestimmten Fällen noch viel schlechter, müssen Gegenmaßnahmen getroffen werden, um dem entgegenzuwirken. [11] Nicht nur ist es schlichtweg unangenehm für den Menschen, wenn Gerüche in der Luft liegen, es kann auch durchaus ungesund sein, sich längerer Zeit gewissen Gasen auszusetzen, die Beiprodukte von Streich-, Putz- oder anderer Arbeiten sind. Solche Gase werden auch VOC (*volatile organic compounds*), zu Deutsch „flüchtige organische Verbindungen“, genannt. [12]

Luftgütesensoren, wie sie auch genannt werden, funktionieren mit Hilfe von Metalloxid-Halbleitergassensoren, die unter Gaseinfluss ihre elektrische Leitfähigkeit verändern. Aus dieser Veränderung des elektrischen Widerstandes lässt sich direkt auf das Vorhandensein

und indirekt auf die Konzentration eines Schadstoffes schließen. [13] Im Kontext einer Hausautomatisierung können die Informationen, die ein solcher Sensor liefert, genutzt werden, um bei der Übertretung eines Schwellenwerts die Fenster zu öffnen oder eine Belüftungsanlage anzuschalten.

1.2.4.2 Aktoren

Wenn Sensoren den Sinnen eines Menschen entsprechen, sind Aktoren (auch *Aktuatoren* genannt) die Körperteile, die vom Menschen gezielt gesteuert werden können, um Aktionen auszuführen, wie die Hände beispielsweise. Sie stellen den zweiten Teil der physikalischen Komponenten dar, ohne sie würde das System zwar wahrnehmen, könnte aber keinen Einfluss auf seine Umwelt haben.

Aktoren können viele verschiedene Formen annehmen, im Grunde kann jedes elektronische Gerät als Aktor fungieren, da mit Hilfe eines Funkempfängers ein *an/aus* Signal empfangen werden kann und somit schon die Voraussetzungen für einen Aktor erfüllt sind. Im Folgenden werden die verbreitetsten Aktortypen beschrieben.

1.2.4.2.1 Lampen

Lichtsteuerung ist mitunter der größte Bereich der Heimautomatisierung und auch einer der ältesten. Lange bevor der Begriff *Smart Home* sich etabliert hatte, hatten Hauseingänge schon von Bewegungsmeldern betriebene Automatisierungen implementiert, die Lampen anschalteten. In solchen Fällen wurden die Schaltungen meist mit elektrischen Leitungen betrieben, die in die Hauswände eingearbeitet waren. Heutzutage gibt es andere Möglichkeiten, Glühlampen zu kontrollieren. Unter anderem gibt es Modelle, die einen Funkempfänger eingebaut haben und so ferngesteuert werden können. Ein Beispiel hierfür ist das Produkt *Hue* der Marke *Philips*. Diese Glühlampen werden beispielsweise mit Hilfe einer sogenannten *Bridge* verbunden, welche dann mit einer Smartphone App gesteuert werden kann. Die Hue Bridge wird über das WLAN angesteuert, genauer durch eine RESTful API Schnittstelle. [14] Die Hue Glühlampen können teilweise auch in unterschiedlichen Farben leuchten, verschiedene Farbtemperaturen annehmen, gedimmt werden uvm. All diese Funktionen sind dementsprechend auch fensteuer- und automatisierbar.

Abbildung 2: Die Philips Hue Produkte [15]

Hue ist nur eins von unzähligen Beispielen, wie Glühlampen ferngesteuert und automatisiert werden können. Um herkömmliche Glühbirnen zu steuern, können auch ferngesteuerte Fassungen verwendet werden oder fengesteuerte Steuereinheiten, die die Stromzufuhr der Glühbirnen kontrollieren.

Lampen sind, wie schon erwähnt, ein klassischer Teil der Hausautomatisierung. Beispielsweise könnte eine Automatisierung das Anschalten einer Lampe beinhalten, wenn der Nutzer nach Hause kommt. Eine andere Möglichkeit ist das Zusammenfassen von mehreren Lampen zu benutzerdefinierten Gruppen, sodass alle Lichter im Wohnzimmer gleichzeitig angehen, wenn der Nutzer den zugehörigen Knopf drückt. Hier sind dem Entwickler keine Grenzen gesetzt.

1.2.4.2.2 Mediengeräte

Ein anderer großer Bestandteil von moderner Hausautomatisierung, besonders im privaten Bereich, ist das Steuern von Mediengeräten aller Art. Dazu gehören z.B. Stereoanlagen oder Fernseher.

Abbildung 3: Der Google Chromecast [16]

Ein solches Mediengerät ist zum Beispiel der *Google Chromecast*. Dieses Gerät wird in den HDMI Port eines kompatiblen Mediengeräts gesteckt und kann daraufhin mit dem Smartphone oder Computer gesteuert werden. So können Apps wie Spotify oder YouTube auf dem TV ge‘castet‘ und bequem gesteuert werden. Chromecast bietet auch die Möglichkeit mit Hilfe einer API (*Application programming interface*, zu Deutsch *Programmierschnittstelle*) eigene Programme zu schreiben und, im Kontext der Hausautomatisierung wichtiger, den Chromecast mit anderen Plattformen zu verbinden und komplexe Automationen zu schreiben. [16]

Ähnlich wie Chromecast funktioniert auch *Apple TV*, eine Set-Top-Box des Herstellers Apple, welche auch per HDMI an den Fernseher angeschlossen wird und daraufhin mit einem Apple-Gerät wie dem iPhone, iPad, oder einer speziellen Apple Remote, gesteuert werden kann. Die Box verfügt über WLAN Empfang und kann somit Medien aus dem Internet von verschiedenen Anbietern streamen. Apple TV lässt sich auch in dritten Plattformen integrieren und mit Hilfe derer steuern. [17]

Abbildung 4: Apple TV [17]

1.2.4.3 Computer

1.2.4.3.1 Raspberry Pi

Der Raspberry Pi ist ein Minicomputer in der Größe einer Kreditkarte, der von der Raspberry Pi Foundation für Bildungszwecke in Schulen und Entwicklungsländern produziert wird.

Basis der Entwicklung des Pis ist die Beobachtung der Hersteller, dass Lernende von Informatikfächern, in Schulen und Universitäten, immer weniger Hintergrundwissen über Computer und deren Funktionsweise haben. Dies wird hauptsächlich der erhöhten Nutzung von "Home PCs" wie Windows oder Macintosh Maschinen oder auch Tablets und Smartphones zugeschrieben. Frühere Generationen lernten das Programmieren auf Produkten wie den Amigas, BBC Micros, Spectrum ZX oder Commodore 64 kennen und waren somit viel näher am eigentlichen Geschehen dran als auf den benutzerfreundlichen Computern von heute. Darüber hinaus wurde der Computer immer mehr zu einem Haushaltsgegenstand, den die ganze Familie nutzte und der essentiell für den Alltag wurde. Herumexperimentieren, was in vielen Fällen Zubruchgehen zur Folge hat, war somit weit weniger attraktiv als bei Maschinen, deren einziger Zweck die Forschung selbst war. Da leistungsstarke Prozessoren immer preisgünstiger wurden, war es nun jedoch möglich Minicomputer im Wert von nicht mehr als 35$ herzustellen. [18]

Angefangen mit dem Raspberry Pi 1 Model B, welcher 2012 herausgebracht wurde, hat die Foundation bisher 6 verschiedene Modelle herausgebracht, darunter auch der Pi Zero, welcher nur 5$ kostet und auch gratis in einer Ausgabe des Pi Magazins zugelegt war. [19]

Der Raspberry Pi hat verschiedene Anwendungsmöglichkeiten [18]:

- Allgemeine Computer Anwendungen: Mit dem Raspberry Pi kann man prinzipiell alles machen, was man mit einem herkömmlichen PC auch machen würde. Natürlich erreicht der Pi aufgrund seiner relativ schwachen Leistung schnell seine Grenzen, jedoch sind Anwendungen wie das Surfen im Web, Office Arbeiten oder auch kleinere Spiele kein Problem.
- Programmieren lernen: Der Raspberry Pi hat im Werkszustand schon Compiler für mehrere Sprachen an Bord, darunter C, Ruby, Python uvm. Zusätzlich installiert sind Programmierumgebungen, die den Einstieg in die Welt der Informatik einfach machen.
- Elektronische Anwendungen: Mithilfe der eingebauten GPIO Pins sind verschiedenste Projekte möglich, die das Erlernen von elektrotechnischen Grundlagen vereinfachen.

- Autonome Produktentwicklung: Die spannendste Anwendung des Pis und auch diejenige, die für diese Arbeit am relevantesten ist.
- Medien-Center: Es gibt inzwischen viele verschiedene Betriebssystem für den Pi, die ihn zu einem vollwertigen Abspielgerät für jegliche Art von Medien macht, sei es Musik, Bilder oder Videos.

Die Raspberry Pi Modelle haben sich über die Jahre weiterentwickelt, jedoch besitzen die modernen Modelle in der Regel fast alle Anschlüsse der alten Versionen. Da der genaue Aufbau aller Versionen über den Umfang dieser Arbeit hinausgeht, wird hier nur auf den der dritten Version eingegangen, also des Raspberry Pi 3 Model B Version 2.

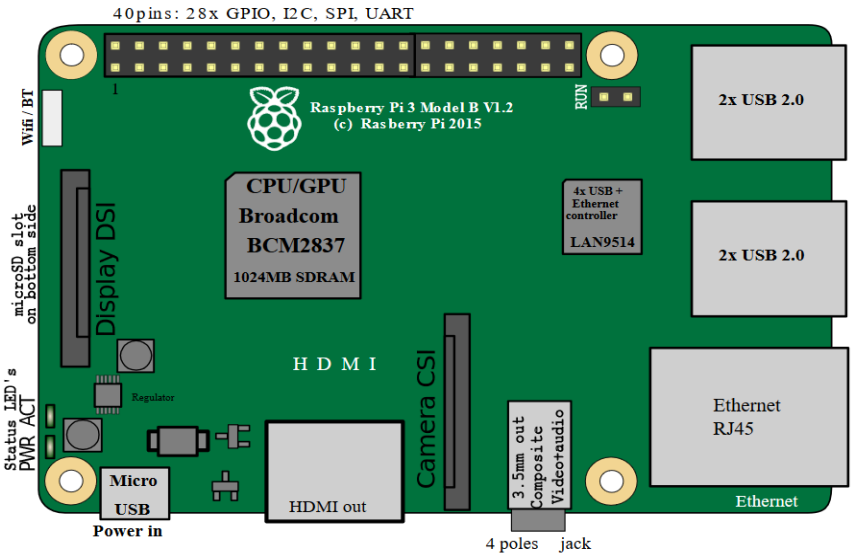

Abbildung 5: Aufbau des Raspberri Pi 3 Model B Version 2 [20]

Dieser besitzt: [21]

- 1GB RAM
- 4 USB Ports
- 40 GPIO Pins
- HDMI Port
- Ethernet Anschluss
- Kombi 3.5mm Audio und Composite Video Anschluss
- Camera Serial Interface Anschluss

- Display Serial Interface Anschluss
- Micro SD Karten Slot
- VideoCore IV 3D Grafik Kern

Darüber hinaus besitzt der Raspberry Pi 3 im Gegensatz zu seinem Vorgänger einen stärkeren Prozessor, einen eingebauten WLAN-Adapter und sowohl ein herkömmliches Bluetooth 4.1 Modul als auch ein Bluetooth Low Energy Modul.

1.2.4.3.2 Arduino

Im Gegensatz zum Raspberry Pi, der im Grunde genommen ein kleiner, günstiger, relativ leistungsschwacher Computer ist, handelt es sich beim Arduino um ein Mikrocontroller Motherboard. Während beim Pi also ein ganzes Betriebssystem installiert ist und man mehrere Programme auf einmal und viele verschiedene Dinge ausführen kann, dient der Arduino dazu ein einzelnes Programm wieder und wieder auszuführen. Das macht den Arduino zu einem weitaus einfacher zu bedienendem Gerät, jedoch ist es gleichzeitig in seiner Funktionalität eingeschränkt. [22]

Arduino entstand 2005 als einfach zu programmierendes Gerät für interaktives Design am Interaction Design Institute in Ivrea, Italien. Die Erfinder wollten etwas entwickeln, was einfach mit verschiedenen Dingen wie Relays, Motoren oder Sensoren zu verbinden, simpel zu programmieren und günstig zu erstehen ist. [23]

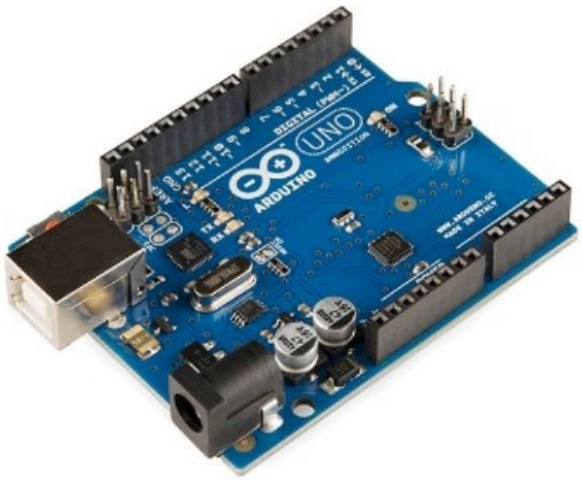

Abbildung 6: Ardiono Uno [24]

Auf der Arduino Webseite gibt es zum Zeitpunkt des Schreibens allein 25 verschiedene Boards und Module zum Verkauf, alles eigenständige Mikroprozessorboards, die für unterschiedliche Zwecke verwendet werden und andere Nischen ausfüllen. [25] Darüber hinaus gibt es einige weitere Boards, die von anderen Produkten abgelöst und daher nicht mehr angeboten werden und unzählige nichtoffizielle mit anderen Arduino Produkten kompatible Boards, mit denen man mehr oder weniger dieselben Ergebnisse wie mit einem Originalprodukt erzielen kann. [26]

Die große Stärke des Arduino, abgesehen von den bisher genannten Aspekten, ist seine integrierte Entwicklungsumgebung (engl.: integrated development environment, IDE), die es für jeden Nutzer sehr einfach macht, Programme für das Board zu schreiben, ohne großes Hintergrundwissen über Mikrocontroller und dergleichen zu haben. Das Board übernimmt den Teil der Übersetzung und Kompilierung in eine Sprache, die der Prozessor versteht. [27]

Beim Coden für den Arduino verwendet man eine Mischung aus C und C++, d.h. in der Arduino Sprache werden eine Sammlung von C/C++ Funktionen implementiert, wobei die Syntax der Sprache C entspricht. Nach dem Transfer des Codes auf das Board wird der „Sketch“ minimal verändert (z.B. das automatische Generieren von Funktionsprototypen) und daraufhin an den C/C++ Kompilierer gegeben. Alle herkömmlichen C/C++ Konstrukte sind vom Kompilierer unterstützt, man ist in der Programmierung also nicht von veralteten Standards eingeschränkt. [28]

Obwohl es durchaus möglich ist einen Arduino für die Verwendung in einem Smart Home Umfeld zu nutzen, eignen sich komplexere Minicomputer wie der Raspberry Pi weitaus besser, da sie natürlich auch eher für diesen Einsatz entwickelt wurden. Besonders in Verbindung mit einer der vielen Software Plattformen zur Entwicklung von Smart Home Anwendungen, stößt der Arduino an seine Grenzen, da er nur für die Ausführung einzelner Skripte geschaffen wurde und nicht für die Ausführung von komplexer Software. Daher wird der Arduino im praktischen Teil dieser Arbeit keine Verwendung finden.

1.2.5 Protokolle

In seiner einfachsten Form ist ein (Kommunikations-) Protokoll eine Vereinbarung zwischen zwei Parteien, wie die zu übertragenden Informationen formatiert werden müssen. In diesem Kapitel geht es spezifischer um Netzwerkprotokolle, die sich auf den Austausch von Informationen oder Daten zwischen Computern bzw. Prozessen beziehen. Für den Austausch von Daten in komplexeren Netzwerken wie dem Internet ist im Normalfall ein Zusammenspiel von mehreren Protokollen notwendig. Zur Strukturierung sind diese Protokolle in Schichten aufgeteilt, die sich einem Protokollturm zusammenfügen. Dieses Referenzmodell heißt auch OSI-Modell und beinhaltet sieben verschiedene Schichten, die alle eng begrenzte Aufgabenfelder besitzen. [29, 30]

Während manche hier vorgestellten Protokolle Teil einer bestimmten Schicht im OSI-Modell sind, stellen andere eher einen eigenen Protokollturm dar und beziehen sich auf mehr als eine Schicht. Das Wort „Protokoll“ ist an dieser Stelle also eher liberal verwendet und bezeichnet nicht eine spezifische Definition des Wortes.

1.2.5.1 Z-Wave

Z-Wave ist ein Kommunikationsstandard, welcher als Protokoll für drahtlose Heimautomatisierungsnetzwerke dient. Es wurde 2001 von der Firma Zen-Sys entworfen und 2008 von Sigma Designs gekauft, welche es heute noch in Form von Hardware und Chips vertrieben. [31, 32] Z-Wave basiert auf einem Lizenzmodell, das effektiv bedeutet, dass alle Hersteller ihr Material von Sigma kaufen müssen und ihre Endprodukte von Sigma testen und bestätigen lassen müssen, bevor sie diese als “Z-Wave” Produkt bezeichnen dürfen. Diese Hersteller haben sich 2005 zur Z-Wave-Allianz zusammengeschlossen und haben zusammen bisher insgesamt ca 2.100 Geräte entwickelt. [33]

Z-Wave nutzt als Kommunikationskanal ein 900 MHz Radiofrequenzsignal, mit einer Übertragungsrate von ungefähr 40 kbit/Sekunde, obwohl modernere Geräte auch teilweise 100 kbit/Sekunde schaffen. Jedes Gerät verfügt ungefähr über eine Reichweite von 20-30

Meter, je nach Gegebenheit der Umgebung – Wände und Decken schränken die Reichweite ein, wie jedes andere RF-Signal auch. Z-Wave Netzwerke fungieren in einer Mesh- oder auch Netzanordnung, was bedeutet, dass die einzelnen Funkmodule Nachrichten an den Empfänger weiterleiten, auch wenn sie eigentlich nicht in der Kommunikationskette vorgesehen sind. So können theoretisch sehr große Netzwerke aufgebaut werden, die dank der Meshanordnung trotzdem voll funktionsfähig, wenn auch durch die größere Zahl an „Hops“, die jede Nachricht zurücklegen muss, langsamer sind.

Jedes Z-Wave Netzwerk besteht aus einem einzigen Controller, der für das gesamte Netzwerkmanagement zuständig ist und bis zu 232 verschiedenen Funkmodulen, unter Z-Wave auch Nodes genannt. Die spezifische Restriktion der Menge an Nodes ergibt sich aus der Adressierung der Geräte (Node ID) und des Netzwerks (Network ID). Der Controller kann entweder alleinstehend sein oder an einen Computer angeschlossen werden, in diesem Fall über den standardmäßigen USB-Port. Außerdem gibt es die Möglichkeit einen Z-Wave Controller direkt an den GPIO Port eines Raspberry Pis anzuschließen.

Die Z-Wave Geräte werden aus Radiofrequenz-Perspektive in zwei Gruppen aufgeteilt: [34]

- Geräte, die in den Schlafmodus gehen können. Alle batteriebetriebenen Geräte besitzen diese Fähigkeit. Zweck des Schlafmodus ist die Einsparung von Energie und die daraus resultierende erhöhte Lebenszeit der Batterien und die allgemeine Stromeinsparung. Diese Geräte können nicht für die Weiterleitung von Nachrichten genutzt werden.
- Geräte, die permanent an bleiben und mit einer „always listening“ flag gekennzeichnet sind. Diese Geräte werden für die Weiterleitung von Nachrichten verwendet, da sie verlässlich sind. Meist sind stromnetzbetriebene Geräte im „always listening“ Modus.

Aus Radiofrequenz-Routing Perspektive gibt es zwei andere Gruppen, in die die Geräte aufgeteilt werden:

- Controller, die, wie schon angesprochen, für das Netzwerkmanagement zuständig sind und Routenzuweisung betreiben. Es gibt pro Netzwerk nur einen Controller.
- Slaves, die wenig oder gar keine Informationen über die Netzwerktopologie besitzen und keine Funktionalität für die Instandhaltung des Netzwerks besitzen (wie das Hinzufügen oder Entfernen von anderen Nodes)

Beim Hinzufügen eines jeden Geräts wird es mit dem Z-Wave Controller gepaart, welcher die Signalstärke zwischen ihnen feststellt, um daraus die Routing Algorithmen zu berechnen. Daraufhin kann das Funkmodul an einer entfernten Stelle platziert werden und vorausgesetzt es befindet sich ein Gerät, das noch dem Netzwerk angehört, in seiner Nähe, kann es mit dem Controller über eine jedes Mal neu berechnete Route kommunizieren.

Das Hinzufügen von Geräten selbst eine der attraktivsten Funktionen von Z-Wave aufgrund seiner Unkompliziertheit: Mit einem Knopfdruck auf dem Z-Wave Controller wird der Pairing Mode aktiviert und mit einem Knopfdruck auf dem Funkmodul wird dieses im Netzwerk aufgenommen. Diese Funktion ist sehr verlässlich und bereitet dem Nutzer nur selten Probleme.

Obgleich seiner proprietären Natur, ist es heutzutage möglich Z-Wave wie einen Open Source Standard zu behandeln und Applikationen auf Basis von Z-Wave Geräten zu schreiben. Allen voran populär ist hier das Open Z-Wave Projekt. Seit 2014 existiert diese Software Bibliothek, die mithilfe von Reverse Engineering entwickelt wurde. Mit diesem Werkzeug ist es möglich viel tiefer in die Funktionsweise von Z-Wave einzublicken und darüber hinaus macht sie das Debugging um ein Vielfaches leichter. [35]

1.2.5.2 ZigBee

ZigBee ist eine weitere Spezifikation für Drahtloskommunikation in den Bereichen Hausautomation, Sensornetzwerke und Lichttechnik. Ziel des Protokolls ist Effizienz im Energieverbrauch und in der Bitrate, zwei Voraussetzungen an Sensornetzwerke. Die Spezifikation ist eine Entwicklung der ZigBee-Allianz, die Ende 2002 gegründet wurde. Sie ist ein Zusammenschluss von derzeit mehr als 230 Unternehmen, welche die weltweite Entwicklung dieser Technologie vorantreiben. 2004 kam die erste ZigBee-Spezifikation auf den Markt. Die mittlerweile unter dem Namen ZigBee 2004 bekannte Version gilt als veraltet und wurde 2006 von einer komplett überarbeiteten Version abgelöst. [36]

ZigBee Netzwerke bestehen aus drei von den Funkmodulen einnehmbaren Rollen: Koordinatoren, Routern und Endgeräten. Ein Netzwerk besteht immer aus genau einem Koordinator und mehreren Routern/Endgeräten [37].

Koordinator: Das mächtigste Gerät im Netzwerk. Von ihm wird das Netzwerk erstellt und die anderen Module treten diesem bei. In vielen Fällen ist der Koordinator das Sammelbecken aller Informationen, die im Netzwerk übermittelt werden. Der Koordinator übernimmt das Verteilen der Kurzadressen.

Router: Sind gewissermaßen Zwischengeräte, die Pakete von Modul zu Modul bringen können. Sie können bereits existierenden Netzwerken beitreten, diese dann mit Hilfe von Beacons “bewerben” und andere Funkmodule dem Netzwerk hinzufügen. Daher haben sie andere Module unter sich, zu denen sie eine Eltern-Kinder Beziehung haben und mit denen sie kommunizieren.

Endgeräte: Die einfachsten Module, sie können weder Pakete weiterleiten noch andere Module unter sich haben und oft betreten sie einen Schlaf-Modus um Energie zu sparen. Ankommender Datenverkehr wird von dessen Elternteil gespeichert bis diese die Daten abrufen. Dadurch können sehr lange Batteriestandzeiten erreicht werden.

ZigBee-Netzwerke können verschiedene Formen annehmen. Je nachdem, wie das Netzwerk vom Koordinator konfiguriert ist, handelt es sich um eine Stern-, Baum- oder Mesh-Topologie. Sternnetzwerke bestehen aus einem Koordinator und sonst nur Endgeräten. Die Netzwerktiefe beträgt somit 1. In Baum- oder Meshnetzwerken kann die vom Koordinator bestimmte maximale Netzwerktiefe erreicht werden (≥ 1). Baumnetzwerke haben eine hierarchische Routingstrategie, da die Adresszuweisung hier der Reihe nach verläuft. Bei Meshnetzwerken funktioniert das Routing mittels Wegentdeckung und Routingtabellen, außerdem unterstützen sie volle Peer-to-Peer Kommunikation unter den Routern.

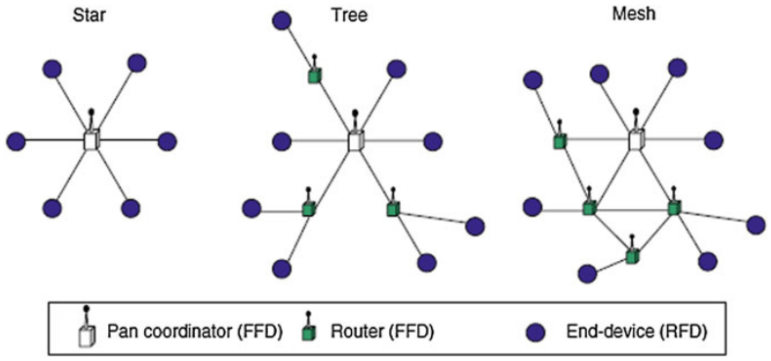

Abbildung 7: ZigBee Netzwerk Topologie [37]

Die ZigBee Architektur basiert auf zwei voneinander unabhängigen Spezifikationen von Referenzmodellen: dem OSI-basierten IEEE 802.15.4 Standard, der die PHY- und die MAC-Schicht bereitstellt und dem von der ZigBee-Allianz spezifizierten Standard, dem die Vermittlungs- (NWK) und Anwendungsschicht (APL) entstammen.

Jede der vier Schichten der ZigBee-Architektur erfüllt bestimmte Aufgaben und stellt für die darüberliegende Schicht verschiedene Dienste bereit. In jeder Schicht gibt es eine Einheit, die für die Dienste der Datenübertragung (data entity) und eine Einheit, die für die restlichen Dienste (management entity), welches insbesondere Management Aufgaben sind, verantwortlich ist. [38]

Die Funktionalitäten der NWK-Schicht beinhalten unter anderem Multihop Routing, Wegerkennung, Sicherheit und das Beitreten und Verlassen eines Netzwerks mit Hilfe von Kurzadressen-Zuweisung (16-bit) für neu beigetretene Module. Außerdem ist sie logischerweise zuständig für das Erstellen der NWK-Frames, also die Teile der übermittelten Pakete, in denen die NWK-relevanten Daten aufbewahrt sind.

Die Netzwerkschicht besitzt eine Reihe für ihre Aufgaben notwendigen Konstanten und Variablen, welche zusammen in der sogenannten Network Layer Information Base (NIB) gespeichert sind, wobei die NIB eher eine abstrakte Sammlung darstellt, als eine wirkliche Datenbank. Fast alle Aktionen eines Funkmoduls benötigen entweder Daten aus der NIB und/oder lösen eine Veränderung der entsprechenden Werte aus.

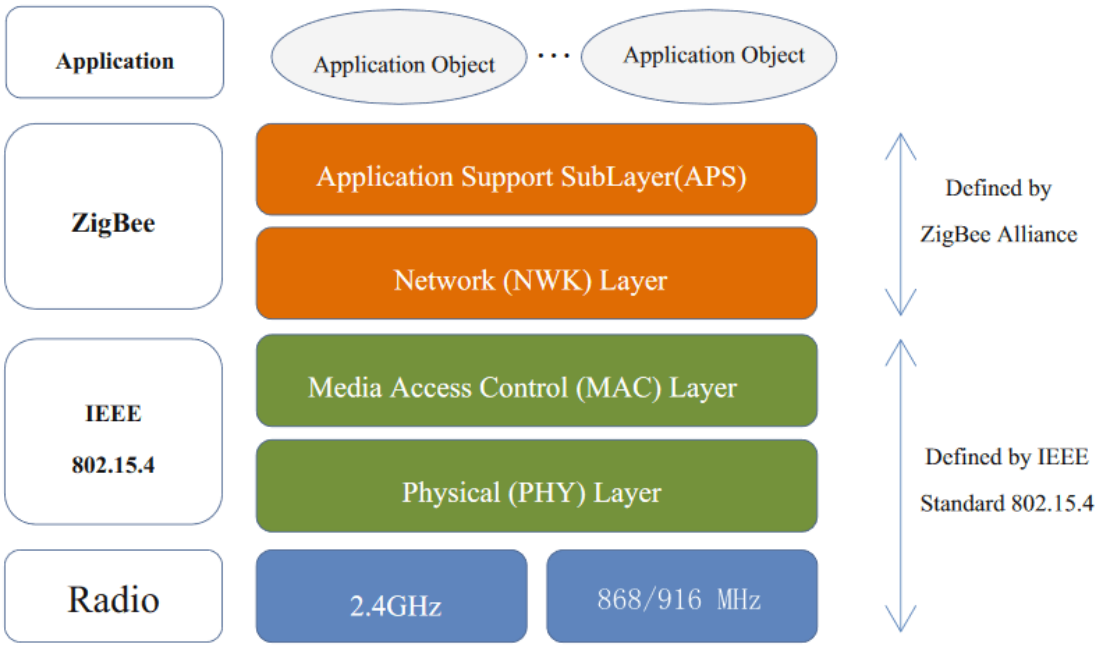

Abbildung 8: Schichten des ZigBee Stacks [38]

Konstanten sind diejenigen Werte, die sich nicht mehr verändern lassen, sobald das Netzwerk in Betrieb genommen wurde. Durch sie werden Grundeigenschaften für das Funkmodul festgelegt, wie z. B. ob das Funkmodul die Fähigkeit hat, als Koordinator eingesetzt zu werden, welche ZigBee-Protokollversion benutzt wird oder welcher Sicherheitsstufe standardmäßig eingesetzt werden soll. [38]

1.2.5.3 BidCoS

Die HomeMatic Produktreihe wird mit dem Funkprotokol BidCoS betrieben. BidCoS steht für „Bidirectional Communication Standard“, was die Funktionsweise des Protokolls bechreibt: Im Gegensatz zu älteren Standards, die die Anfänge der Smart Home Automatisierung darstellten, funktioniert BidCoS mit Sende- und Empfangsbestätigungen, welche eine sehr hohe Sicherheit in Bezug auf die Funktion des Systems ermöglicht.

Bei BidCoS handelt es sich um ein ausschließlich proprietäres Protokoll, welches im geistigen Eigentum der HomeMatic Herstellerfirma eQ-3 steht. Daher stehen nicht viele Informationen über die Funktionsweise von BidCoS zur Verfügung, außer seiner bidirektionalen Natur. Ein weiterer Nachteil daran, dass das Protokoll proprietär ist, ist der vergleichsweise hohe Preis, den der Kunde für die HomeMatic Geräte zahlen muss. Nichtsdestotrotz sind die Systeme meist sehr zuverlässig und haben einen sehr großen Umfang.

Über das mit BidCoS betriebene HomeMatic System können komplette Gebäudesteuerungen realisiert werden. Dazu gehören zum einen sicherheitsrelevante Dinge, wie Zugangskontrolle oder Einbruchssicherung, aber auch Schutz vor Gefahren durch Gas, Wasser und Rauch. Darüber hinaus ist es möglich die Klimatisierung des Hauses (Heizungen, Lüftung, Kühlung) mit HomeMatic umzusetzen, wie auch Licht oder andere elektrische Verbraucher. HomeMatic Systeme können sowohl batteriebetriebene als auch netzabhängige Komponenten beinhalten. [39]

1.2.5.4 Bluetooth LE

Bluetooth Low Energy (auch Bluetooth Smart) ist ein Protokoll für drahtlose Netzwerkkommunikation mit relativ geringer Reichweite. BLE wurde zunächst als Bluetooth 4.0 entwickelt, hat jedoch in Wirklichkeit eine andere Abstammung und andere Design Ziele als das klassische Bluetooth. BLE wurde zunächst von Nokia als Wibree vermarktet, bevor es von der Bluetooth Special Interest Group (SIG) übernommen und mit dem Bluetooth Standard verbunden wurde. [40]

Im Gegensatz zu anderen Standards im Bereich Personal Area Network, wurde BLE nicht als Gesamtlösung für Drahtlosnetzwerke entworfen, sondern als Standard für Radiofunkkommunikation mit dem möglichst niedrigstem Energieverbrauch, um die Vorsätze „low cost, low bandwidth, low power, low complexity" einzuhalten und so gut wie möglich zu erreichen. [41] Die Geschichte des klassischen Bluetooths zeigt eine fortdauernde Steigung der Bitrate, was an der Anwendung des Protokolls liegt. Während es ursprünglich als bloße Kommunikation zwischen Computern und Mobilfunkgeräten (Laptop und Handy) geplant war, stellte sich später heraus, dass es hauptsächlich für das Nutzen von drahtlosen Headsets verwendet wurde. Mit zunehmender Zeit kamen noch mehr Anwendungen dazu, wie drahtloses Drucken von Dokumenten, Datei Transfers oder Musik Streaming. Jeder dieser Anwendungsfälle benötigte mehr Bandbreite und daher steigerte sich die Bitrate des Protokolls mit jeder neuen Version. Während der erste Bluetooth Entwurf noch eine Datenrate von 1 Megabit pro Sekunde (Mbps) besaß, konnte Bluetooth 3.0 schon mehrere *Hunderte* Mbps übertragen. [42]

Modems	Ethernet
V.21: 0.3kbps	802.3i: 10Mbps
V.22: 1.2kbps	802.3u: 100Mbps
V.32: 9.6kbps	802.3ab: 1000Mbps
V.34: 28.8kbps	802.3an: 10000Mbps

Wi-Fi	Bluetooth
802.11: 2Mbps	v1.1: 1Mbps
802.11b: 11Mbps	v2.0: 3Mbps
802.11g: 54Mbps	v3.0: 54Mbps
802.11n: 135Mbps	v4.0: 0.3Mbps

Abbildung 9: Geschwindigkeiten nehmen normalerweise zu [42]

Bluetooth Low Energy dagegen geht in eine ganz andere Richtung. Anstatt die Datenrate zu erhöhen, wurde es für den kleinstmöglichen Energieverbrauch optimiert, was zur Folge hat, dass die Datenrate nicht sonderlich groß ist. Interessant wird dies, wenn man in Betracht zieht, dass andere drahtlose Kommunikations- protokolle ihre Geschwindigkeiten stetig erhöhen, wie in Abb. 9 zu sehen ist.

Da das ultimative Ziel von BLE das Einsparen von Kosten ist, ergibt es Sinn, dass Geräte, die das Protokoll verwenden, in ihrer Bauart sehr minimalistisch und kostensparend sind. Die Größe der Batterie ist ein weiterer Punkt, an dem Einsparungen möglich sind. Die logische Folge ist also, kleinere Batterien zu verwenden, wie die Knopfbatterie. Bluetooth Low Energy hat sich als eine der wenigen Technologien herausgestellt, die realistisch mit einer solchen Batterie einen Monat oder auch länger betrieben werden kann, da die Energie so effizient genutzt werden kann. [42]

1.2.5.5 MQTT

MQTT ist ein Nachrichtenprotokoll, das erstmals 1999 von IBM und Eurotech entworfen wurde. [43] Es ist ein extrem einfacher und leichtgewichtiger Standard, der auf der Publish/Subscribe Architektur basiert. MQTT eignet sich am besten zur Nutzung in eingeschränkten Netzwerken, wo die Bandbreite klein oder die Wartezeit hoch ist und/oder die Funkmodule wenig Leistung in Form von Prozessorstärke und Speicher haben. Wie auch HTTP oder SMTP befindet sich MQTT in der Applikationsschicht des OSI-basierten IEEE 802.15.4 Standards und liegt somit auf den Schichten TCP/IP.

MQTT Netzwerke bestehen aus zwei Arten von Komponenten: Clients und dem Message Broker. Clients sind alle Endgeräte in einem Netzwerk, die MQTT „sprechen“ und Informationen austauschen wollen. Dabei ist jeder Client immer nur mit dem Broker verbunden. Diesem teilt er seine „Interessen“ in Form von Abonnements und Publishes mit. Dem Broker kommt dann die Aufgabe zu, diese Interessen zu verwalten. Der Broker ist das Herz eines jeden Netzwerkes, darum existiert in jedem auch nur ein einziger. [44]

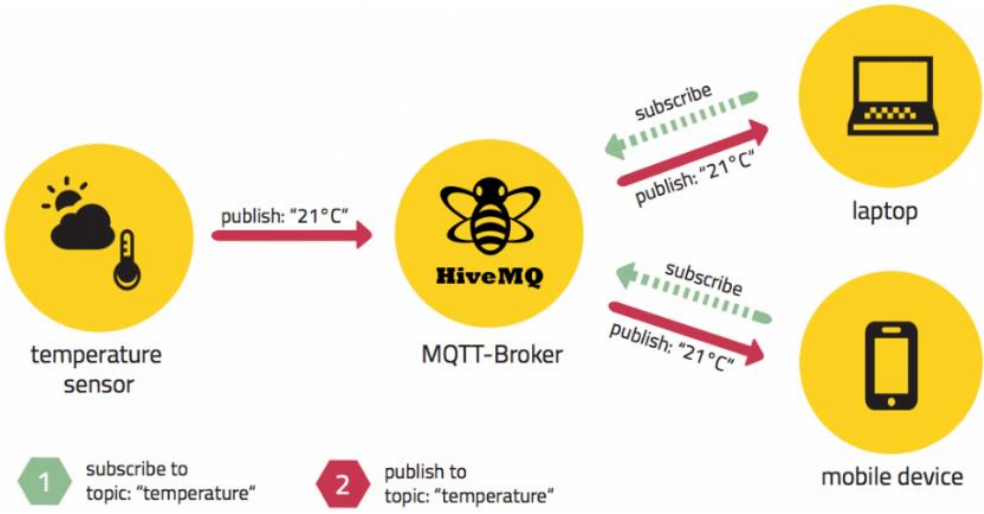

Abbildung 10: Exemplarisches MQTT Netzwerk mit Publish/Subscribe Struktur [45]

MQTT basiert auf einigen Kernkomponenten, die sicherstellen, dass die Nachrichten sicher ankommen und gleichzeitig so leichtgewichtig wie möglich sind [46]:

- Publish/Subscribe & Topics/Subscriptions: Nachrichten in einem MQTT Netzwerk werden unter spezifischen Themen (*topics*) veröffentlicht, welche Teilnehmer des Netzwerks abonnieren (*subscribe*) können, dargestellt in Abb. 10. So wird sichergestellt, dass jeder Teilnehmer nur die Nachrichten bekommt, die er auch haben will. Ein weiterer Vorteil des Modells ist die reduzierte Größe der Message Header, da nicht jedes Endgerät adressiert werden muss, sondern viel mehr ein Mailverteiler adressiert wird.
- Quality of Service (*QoS*) Levels: Das MQTT Protokoll spezifiziert drei unterschiedlich starke Prioritäten für Nachrichten, von QoS 0 (*At most once*), das am wenigsten Aufwand einher bringt über QoS 1 (*At least once*) bis hin zu QoS 2 (*exactly once*), das am meisten Aufwand verlangt. Je höher das QoS Level, desto zuverlässiger die Zustellung, aber auch desto höher die Netzwerkauslastung.
- Aufbewahrte Nachrichten: Auf Englisch „*retained messages*“ werden vom Message Broker an jeden neuen Abonnenten geschickt, sobald dieser beitritt. So wird sichergestellt, dass neue Teilnehmer nicht zunächst auf die nächste Publikation warten müssen für wichtige Informationen.
- Clean sessions und beständige Verbindungen: Ist die *clean session* für einen Teilnehmer gesetzt, so wird er vom Netzwerk vergessen, sobald er einmal die Verbindung verliert. Ist sie nicht gesetzt, ist seine Verbindung beständig

(*durable*) und nach Verbindungsabbrüchen bleiben seine Abonnements bestehen und Nachrichten mit hohem QoS Level werden ihm aufbewahrt.

MQTT findet heutzutage schon sehr viel Verwendung in zahlreichen Projekten von Firmen wie Facebook, Amazon oder Microsoft. Obgleich seiner passenden Funktionalität wird das Protokoll nicht nur für IoT Anwendungen genutzt, es lässt sich grundsätzlich in allen Bereichen verwenden, wo die Verbindung langsam oder unzuverlässig ist. [47]

1.2.5.6 HTTP und REST

HTTP ist ein zustandsloses Netzwerkprotokoll, welches sich auf der Anwendungsschicht des Internetprotokollturms befindet. HTTP wird hauptsächlich zur Übertragung von Daten von Webservern durch Webbrowser auf andere Maschinen genutzt und bildet das Rückgrat des modernen Internets. Das Protokoll wurde ursprünglich von Tim Berners-Lee im Jahre 1991 entwickelt, als ein erster Prototyp vervollständigt wurde, welcher nachträglich den Namen HTTP 0.9 bekam. [48]

Die Funktionsweise des Protokolls basiert auf Anfragen an den Server, woraus sich auch die Zustandslosigkeit ergibt: Der Client, ein Webbrowser in den meisten Fällen, schickt eine Anfrage an den Webserver, woraufhin der eine Antwort liefert, die im klassischen Szenario aus einem Kopfteil und dem Inhalt einer Webseite besteht. Mit zunehmender Komplexität des Internets wurden die Anwendungsfälle, die mit HTTP-Requests verbunden sind, vielfältiger, sodass eine Vielzahl von HTTP Anfragemethoden (vgl. Tabelle 2) geschaffen wurde, auch Verben genannt. [49]

GET	Gängigste Methode. Forderte eine Ressource unter Angabe eines URI vom Server an
POST	Sendet eine Sammlung von Argumenten an die spezifizierte Ressource und führt oft zu einer Änderung des Zustands oder anderen Nebeneffekten
PUT	Ersetzt die spezifizierte Ressource komplett mit den mitgesendeten Daten
DELETE	Löscht die angegebene Ressource

Tabelle 2: HTTP Verben

Wenn im Kontext des Internet of Things über HTTP geredet wird, dann meist in Verbindung mit REST (*Representational state transfer*). Ziel einer RESTful API, wie sie auch genannt wird, ist die Nutzung der HTTP Werkzeuge im Sinne ihrer Erstellung. REST ist an sich nur ein Konzept und ist vollkommen davon abhängig, wie es in einem realen Webservice umgesetzt wird. Webserver können auch auf andere Art und Weise programmiert werden. Viele Webservices heutzutage ignorieren die Spezifikationen der HTTP Methoden und benutzen nur GET und/oder POST Anfragen. Diese Herangehensweise ignoriert die Grundlagen des HTTP Protokolls und arbeitet im schlimmsten Fall gegen sie. [50] REST ist die empfohlene Architektur für Webservices, da sie sich am besten an die ursprüngliche Architektur von HTTP anbindet.

Ein weiterer wichtiger Aspekt von REST ist, dass Ressourcen *Repräsentationen* von Objekten sind (daher auch der Name). Insofern stellt eine Ressource immer den aktuellen Zustand eines Objekts dar und sollte nicht mit dem Objekt selbst verwechselt werden. Eine RESTful API spezifiziert für die verschiedenen Anfragemethoden auf bestimmte Pfade bzw. Ressourcen jeweils eine auszuführende Aktion. Diese Aktionen beschränken sich in den meisten Fällen auf das Spektrum von CRUD Aktionen, also *Create, Read, Update, Delete.* So könnte beispielsweise eine `GET` Anfrage an die Ressource `/books/123` dem Benutzer das Buch (bzw. alle vorhandenen Daten, die zur Buch Repräsentation gehören) der ID „123“ zurück liefern. Eine POST Anfrage an die Ressource `/books/` würde hingegen eine neue Ressource des Typs „book“ anlegen, wofür die Daten, die im POST Request übermittelt wurden, verwendet werden.

GET	/v1/me/following/contains	Check if User Follows Users or Artists
PUT	/v1/users/{owner_id}/playlists/{playlist_id}/followers	Follow a Playlist
DELETE	/v1/users/{owner_id}/playlists/{playlist_id}/followers	Unfollow a Playlist
PUT	/v1/me/tracks?ids={ids}	Save tracks for user

Abbildung 11: Auszug aus der Spotify REST Api [51]

1.2.6 Plattformen

Im folgenden Kapitel sollen einige Plattformen zur Smarthome Entwicklung vorgestellt werden, indem Geschichte, Technik und weitere Aspekte beleuchtet werden. Außerdem sollen Kriterien wie Verbreitung, technische Umsetzung oder Level von Support dazu dienen, diejenigen Plattformen auszusuchen, mithilfe derer die Umsetzung des geplanten Szenarios passieren wird.

Die Verbreitung wird anhand von feststellbaren Größen gemessen: Anzahl der Suchergebnisse auf Plattformen wie GitHub, Google oder reddit. Außer Frage steht, dass diese Größen mitunter sehr irreführend sein können, darum dienen sie eher als Orientierungshilfe. Für GitHub wird die Metrik der Zahl der Repositories verwendet. Während auch die Zahl der Code-Snippets, in denen der Suchbegriff vorkommt, oder Commits bei der Suche aufgelistet wird, geben diese weniger Auskunft, über die Popularität der Plattform.

Normalerweise eignet sich die Anzahl an Ergebnissen auf der Webseite *Stackoverflow* sehr gut, um die Verbreitung einer Software zu überprüfen. Jedoch verfügen drei der vier Plattformen über eigene Hilfsforen, womit ein solcher Vergleich wenig sinnvoll ist.

1.2.6.1 Home-Assistant

Home-Assistant ist eine Open Source Home Automation Plattform, die erstmals Ende 2013 veröffentlicht wurde. Der ursprüngliche Entwickler, Paulus Schoutsen, hatte das Ziel sein Haus selbst zu automatisieren, basierend auf der Programmiersprache Python. Das System sollte in der Lage sein:

- Den Router abzufragen, welche Geräte verbunden sind
- Die Lichter anzuschalten, wenn die Bewohner nach Sonnenuntergang heimkommen
- Die Lichter abzuschalten, wenn niemand im Haus ist, die Lichter aber noch an sein sollten

Nachdem das alles funktionierte, bemerkte er, dass es während der Dämmerung dunkler wird, also entwickelte er eine Funktion, um die Lampen zu dimmen bevor die Sonne vollständig untergeht. Mehr und mehr Funktionen wurden hinzugefügt und schließlich entschloss sich Schoutsen das Programm der Öffentlichkeit zugänglich zu machen. [52]

Das ist nun fast drei Jahre her [53] und inzwischen ist Home-Assistant bei Version 0.47 angekommen und hat unzählige weitere Funktionen bekommen und unterstützt inzwischen

den sehr großen Teil der heutigen Smart Home Landschaft. Auf der Home-Assistant Webseite sind insgesamt 713 unterstützte Komponenten aufgelistet, die auf der Plattform funktionieren und integriert werden können. [54] Unterstützung für diese Geräte und Systeme wird hauptsächlich von der Community bereitgestellt, da Home-Assistant auf GitHub von jedem Nutzer selbst weiterentwickelt werden kann und auch die Dokumentation dort editierbar ist. [55]

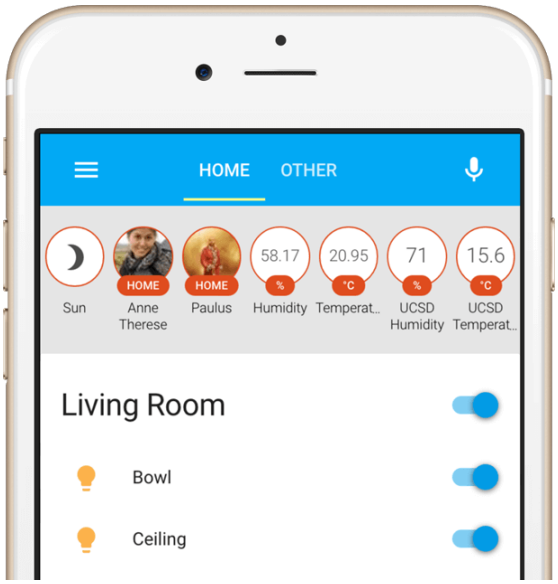

Abbildung 12: Responsives Frontend [56]

Das System stellt dem Nutzer ein responsives Frontend-UI zur Verfügung, welches über einen herkömmlichen Browser zugänglich ist. [57] Dadurch wird die Bedienung erleichtert, auch wenn die Einrichtung nicht sehr simpel ist und einiges an programmiertechnischem Wissen voraussetzt. Auch eine iOS-App wurde schon entwickelt, welche das iPhone dann auch direkt in das Home-Assistant Umfeld einbindet, wodurch Sachen wie Batteriestand oder Anwesenheit im WLAN Netzwerk auch vom Computer aus überprüft werden können. [58]

1.2.6.1.1 Verbreitung

Die Recherche nach der Verbreitung gestaltet sich bei Home-Assistant schwierig, weil die Rechtschreibung des Namens nicht eindeutig ist. Im Code und Repositories werden die beiden Wörter für gewöhnlich mit einem Bindestrich verbunden, so auch in der URL der Webseite (home-assistant.io), jedoch ist das wohl eher technisch bedingt, da Leerzeichen in Programmiersprachen immer substituiert werden müssen, entweder durch Bindestriche, Unterstriche oder Camelcasing. Im Fall von Home-Assistant werden die Wörter auch oft

einfach zusammengeschrieben (homeassistant), besonders als Name von Repositories. Die Aussagekraft der Zahlen ist also besonders in diesem Fall mit Vorsicht zu betrachten. GitHub liefert für den Begriff „home-assistant“ 1.376 Ergebnisse für Repositories. [59]

Auch die Zahl der Google Ergebnisse muss sehr kritisch betrachtet werden. „Home“ und „Assistant“ sind leider sehr generische Begriffe, eine unangepasste Suche nach „Home Assistant“ (ohne Anführungsstriche) liefert beispielsweise 736 Millionen Ergebnisse, weil auch Treffer gezählt werden, die nur den Begriff „Home“ oder nur den Begriff „Assistant“ beinhalten. [60]

Um die Suche zu verfeinern, kann man die Anführungsstriche um die beiden Begriffe packen, damit nur Ergebnisse angezeigt werden, die beide Begriffe beinhalten. Leider wird auch die Kombination „Home Assistant“ sehr häufig im Kontext Smarthome verwendet, auch wenn es nicht um die hier vorgestellte Plattform geht. Beispielsweise kommt schon bald nach den ersten Ergebnissen, die noch die Plattform behandeln, Suchergebnisse zu Googles Home Assistenten „Google Home“ oder auch zu Ausschreibungen für Stellen als Pfleger in „Care Homes“ (Pflegeheimen). Eine solche Suche ergibt 475.000 Ergebnisse. [61]

	Gesamt	Letzte 7 Tage	Letzte 30 Tage
Themen	12.100	304	1.100
Beiträge	114.600	3.100	11.700
Nutzer	9.300	203	818

Tabelle 3: Aktivitätsstatistik des Home-Assistant Forums

Eine eingeschränkte Suche, die die größte Menge an *false positives* ausschließt, wäre beispielsweise *"home assistant" -google -nursing -nurse* und liefert 357.000 Ergebnisse. [62] Auch hier findet man noch falsche Ergebnisse, aber die Suche weiter einzuschränken verändert das Ergebnis nur noch marginal.

Das eigene Hilfsforum der Plattform ist die eigentliche Anlaufstelle für technische Fragen und alle anderen Anliegen rund um Home-Assistant. Das Forum verrät die folgenden Allzeit Statistiken: [63]

Im Laufe der Recherche zum Thema Smart Home auf der Webseite *reddit* ist außerdem die subjektive Wahrnehmung entstanden, dass auch in anderen Communities, die das Thema Heimautomatisierung behandeln, wie „homeautomation“ oder „smarthome“, sehr oft Home-Assistant behandelt und auch empfohlen wird. Zumindest auf dieser Plattform wird das Bild vermittelt, dass Home-Assistant sich zum Spitzenreiter in der Kategorie der Smarthome Software Plattformen entwickelt hat.

Zusätzlich bietet die Home-Assistant Community einen Live Chat, der sich reger Aktivität erfreut. Ursprünglich befand sich dieser auf der Plattform „gitter“, welche den großen Vorteil der GitHub Integration erfreut, jedoch wurde seither ein Umzug auf die Chat Software „Discord“ vollzogen. [64] In beiden Fällen haben diese direkten Kommunikationskanäle den Vorteil der schnelleren Kommunikation und der daraus folgenden, schnellen Lösung von Problemen. Zum Zeitpunkt des Schreibens befanden sich 193 Nutzer und 4 Moderatoren im Discord Kanal. Wie im Fall von reddit ist auch hier der riesige Vorteil gegeben, dass man mit Entwicklern des Projekts kommunizieren kann und somit sehr zuverlässige Antworten erhält.

1.2.6.2 FHEM

FHEM ist eine deutsche Open Source Hausautomatisierungsplattform, die erstmals 2005 veröffentlicht und vom Entwickler Rudolf König erfunden wurde. Verglichen mit den anderen Plattformen existiert FHEM schon weitaus länger. Damals fing König damit an, als er seine Heizungssteuerung „erneuern“ wollte und schnell an Grenzen gestoßen ist. Er programmierte daraufhin eine Plattform in der Programmiersprache Perl, die er bald darauf auch online stellte, woraus sich schnell eine ganze Community von Entwicklern etablierte, die das Projekt weiterbauten. [65] Inzwischen ist FHEM bei Version 5.8 angelangt und wird kontinuierlich weiterentwickelt.

FHEM wird standardmäßig in einem Webbrowser angesteuert, wo seine Benutzeroberfläche verfügbar ist, verfügt aber auch sowohl über eine iOS App namens „FHEMApp“ [66] als auch eine Android App namens „AndFHEM“ [67], welche beide nicht *offiziell* sind, was jedoch Teil des Open Source Modells ist.

1.2.6.2.1 Verbreitung

Für FHEM ist die Recherche nach der Verbreitung etwas schwierig, da es sich hier um ein sehr deutschlandzentriertes Projekt handelt und der Großteil der Ressourcen und Erwähnung auf der FHEM.de, dem FHEM Forum oder dem eigenen Wiki stattfindet. Source Code Austausch findet außerdem nicht auf GitHub statt, wie es sehr üblich ist bei

englischsprachigen Softwareprojekten, sondern auf einer eigenen Repository Alternative namens SVN. Das SVN stellt leider keine Statistiken bereit, darum ist es nicht möglich eine wertvolle Aussage über die Menge von Community generiertem Code zu machen. Die Suche nach dem Begriff „FHEM“ auf GitHub liefert 411 Ergebnisse für Repositories. [68]

In Google erreicht man mit dem Begriff „FHEM“ ungefähr 479.000 Ergebnisse. [69]

Das eigene Hilfsforum der Plattform ist die eigentliche Anlaufstelle für technische Fragen und alle anderen Anliegen rund um FHEM. Das Forum verrät die folgenden Allzeit Statistiken: [70]

- 639455 Beiträge
- 66011 Themen
- 18419 Mitglieder

1.2.6.3 OpenHAB

openHAB ist eine Open Source Hausautomatisierungs- plattform, die erstmals 2010 vom Entwickler Kai Kreuzer veröffentlicht wurde. Ursprünglich wollte Kreuzer eine Software entwickeln, um die Geräte in seinem Haus zu steuern. Daraus entwickelte sich dann ein sehr großes und noch heute weiterentwickeltes Projekt. openHAB ist in Java programmiert und wird seit 2013 als offizielles *Eclipse* Projekt unter dem Namen *Eclipse SmartHome* betrieben. Ziel von Eclipse SmartHome ist die Entwicklung von professionellen Smart Home Lösungen basierend auf der Grundlage des Eclipse Projektes. Somit handelt es sich bei Eclipse SmartHome und ein Framework, auf dem verschiedene Projekte basieren, darunter auch openHAB.

openHAB wird standardmäßig in einem Webbrowser angesteuert, wo seine Benutzeroberfläche liegt, verfügt aber auch sowohl über eine iOS App [71], als auch eine Android App. [72]

1.2.6.3.1 Verbreitung

Der Suchbegriff „openHAB“ erreicht auf GitHub 1.330 Ergebnisse für Repositories. [73]

Auf Google werden für den Suchbegriff „openHAB“ 394.000 Ergebnisse zurückgeliefert. [74]

Das eigene Hilfsforum der Plattform ist die eigentliche Anlaufstelle für technische Fragen und alle anderen Anliegen rund um openHAB. Das Forum verrät die folgenden Statistiken: [75]

	Gesamt	Letzte 7 Tage	Letzte 30 Tage
Themen	15.700	165	801
Beiträge	149.800	2.300	9.700
Nutzer	14.200	168	703

Tabelle 4: Aktivitätsstatistik des openHAB Forums

1.2.6.4 Node-RED

Node-RED ist ein "flow-basiertes" Programm für die Programmierung von Anwendungen und Services im Internet der Dinge. Es wurde ursprünglich von IBM entwickelt, jedoch ist es seit 2016 Teil des JS Foundation Projektes, welches es zu einem Open Source Projekt macht. Node-RED ist in JavaScript geschrieben und basiert auf einer Node.js-Laufzeitumgebung, welche in einem Webbrowser aufgerufen wird. In Node-RED werden per Drag&Drop sogenannte „Nodes" miteinander verbunden, woraus Programmierabläufe entstehen, die mit einem Mausklick wieder auf das System installiert werden.

1.2.6.5 Auswahl der Plattformen

Während Home-Assistant, openHAB und FHEM explizit für die Entwicklung von Smart Home Projekten entwickelt wurden, handelt es sich bei Node-RED um eine Anwendung, die allgemeiner als „Internet of Things Programmierumgebung" bezeichnet werden sollte. Zwar ist es durchaus möglich Projekte in dieser Plattform umzusetzen, jedoch wird die Anbindung von komplizierteren Systemen sehr schwierig und birgt sehr viele Fehlerquellen. Im Endeffekt handelt es sich bei Node-RED nicht um eine Smart Home Plattform, weshalb ein Vergleich auch wenig sinnvoll wäre. Die Plattform unterscheidet sich schlichtweg viel zu sehr von den anderen drei, die sehr ähneln und daher besser geeignet sind verglichen zu werden. Daher wird das Szenario nicht in Node-RED umgesetzt.

2 Ein exemplarisches Smart Home Szenario

Zur Demonstration der Funktionsweise der Plattformen und um den Zweck und Aufbau eines Smart Homes zu simulieren, soll nun ein beispielhaftes Szenario entwickelt werden, wie es in einem modernen Haus implementiert werden könnte. Das Beispiel soll möglichst viele brauchbare Technologien umfassen und dabei ein Großteil der Bereiche abdecken, die ein Smart Home ausmachen.

Da es heutzutage unzählige Anwendungsmöglichkeiten für das Smart Home gibt und man die Möglichkeit riesige Automation zu schreiben und hunderte von verschiedenen Sensoren zur Verfügung hat, muss für diese Arbeit ein sinnvoller Rahmen gespannt werden. Zwar soll die Beispielanwendung noch realistisch sein und einige der populären Technologien verwenden, gleichzeitig gilt es jedoch darauf zu achten sich nicht zu übernehmen, da die Komplexität eines jeden Projektes exponentiell auch die Fehleranfälligkeit und Wartungsmühen in die Höhe schnellen lässt.

Dieser Aufbau soll dann später so gut wie möglich in drei der vorgestellten Plattformen umgesetzt werden und aus der Implementierung dann ein Urteil darüber entstehen, welche Plattform sich am besten eignet.

2.1 Aufbau mit notwendigen Sensoren, Aktoren

2.1.1 Szenario

Dieses Projekt simuliert eine Art Überwachungsanlage und automatisierte Lichtschaltung. Die Funktionalität kann folgendermaßen beschrieben werden:

Es werden zwei Fälle unterschieden – entweder der Nutzer ist daheim oder nicht. Je nach dem was der Fall ist, werden verschiedene Funktionen ausgeführt. Ist der Nutzer daheim und der Bewegungssensor nimmt eine Bewegung wahr, dann wird das Licht angeschaltet. Ist der Nutzer nicht daheim und der Sensor registriert Bewegungen, so wird eine Mail versandt, in der auf die Bewegungen aufmerksam gemacht wird.

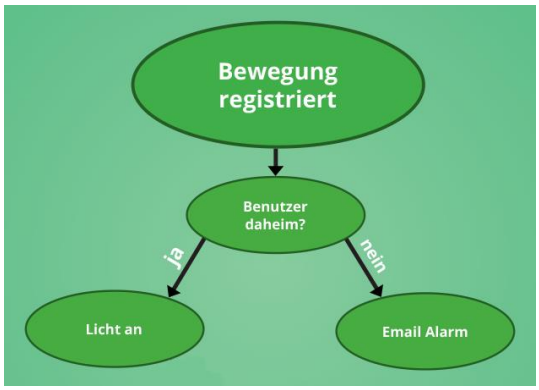

Abbildung 13: Das Szenario

Damit deckt das Szenario die wichtigsten Teile einer Automation ab: Einen Auslöser, eine Kondition und zwei Aktionen, die je nachdem, ob die Kondition erfüllt ist, eintreten. Außerdem müssen Sensorinformationen ausgelesen und überprüft werden.

Für die Umsetzung des Szenarios werden verschiedene Sensoren, Aktoren und anderes Equipment benötigt:

- Bewegungsmelder
- Temperatursensor
- Dimmer
- Rollladensteuerung
- Präsenzerkennung

Für diese Zwecke wurden, nebst einem Raspberry Pi Model B Version 2, auf den schon im Hardware Kapitel eingegangen wurde, die folgenden Gerätschaften verwendet:

2.1.2 Aeon Lab Z-Stick S2

Abbildung 14: Z-Stick S2 [76]

Der Z-Stick ist das Herzstück des Z-Wave Netzwerks und somit unabdingbar. Hierbei handelt es sich um eine Art USB-Stick, der in den Kerncomputer gesteckt wird. An ihn senden die Z-Wave Komponenten ihre Nachrichten und dieser stellt sie dann der Software bereit. Der Z-Stick ist mit jedem Gerät kompatibel, das den Z-Wave Standard verwendet, was die Nutzung von vielen verschiedenen Sensoren und Aktoren vereinfacht.

Die einzige Möglichkeit den Stick mit der Hand zu bedienen ist mit der Z-Wave Taste, die zum Hinzufügen und Entfernen von Funkmodulen da ist und verschiedenen anderen Netzwerkmanagement-Funktionen.

Der Z-Stick hat eine Reichweite von 50 Metern [77], welche eine höhere Reichweite eines Sensors insofern nichtig macht, als dass das Netzwerk nicht funktionieren kann, wenn der Z-Stick den Funkmodulen keine Nachrichten schicken kann. Insofern hat das Netzwerk des S2 einen maximalen Radius von 50 Metern.

2.1.3 Aeon Lab Multisensor Gen5

Abbildung 15: MultiSensor Gen5 [78]

Der MultiSensor hat vier Sensoren eingebaut: Bewegung, Temperatur, Luftfeuchtigkeit und Licht. Der MultiSensor funktioniert mit dem Z-Wave Plus Protokoll, was ein Upgrade des standardmäßigen Z-Wave Protokoll ist. Das Gerät lässt sich einfach an Wänden und Decken montieren und läuft entweder mit Batterie oder externem Strombetrieb via MiniUSB-Anschluss. Im Batteriebetrieb geht der Sensor in einen Schlafmodus, bei dem nur wenige Nachrichten übermittelt werden, um Strom zu sparen. [79]

Der MultiSensor verfügt über die folgenden technischen Spezifikationen: [80]

- Temperatur-Reichweite: -10 - 50°C
- Temperatur-Genauigkeit: ±1°C
- Luftfeuchtigkeit-Reichweite: 20% - 80%
- Luftfeuchtigkeit-Genauigkeit: ±5% (bei 25°C)
- Lichtmessung-Reichtweite: 1 - 1000 LUX
- Bewegungsmeldung-Empfindlichkeit: 3 - 5 Meter
- Reichweite: bis zu 150 Meter

2.1.4 Happy Bubbles Presence Detektor

Abbildung 16: HappyBubbles Presence Detector [81]

Hierbei handelt es sich um eine Bluetooth WLAN Schnittstelle. Der Detektor horcht auf Low Energy Bluetooth Signale von Geräten, die ihre Anwesenheit bewerben. Ein solches Gerät könnte ein so konfiguriertes Smartphone oder ein so genannter Bluetooth Beacon sein. Der Detektor wird zum WLAN Netzwerk hinzugefügt und kommuniziert per MQTT mit dem MQTT Broker.

Der Happy Bubbles Detektor besteht aus einem NodeMCU Chip, einem WLAN und einem Bluetooth Modul. Da MQTT über WLAN funktioniert, ist sonst nichts notwendig, um die Funktionalität des Detektors zu ermöglichen. [82]

Zweck des Geräts ist eigentlich die Aufenthaltsorterkennung einer Person im Haus. Dies wird mit mehreren Exemplaren erreicht, welche sich alle in verschiedenen Räumen befinden. Läuft die Person mit dem Beacon durch das Haus, erkennt jeder der Detektoren wie weit sie

entfernt ist. Basierend darauf, an welchem Detektor sie am nächsten dran ist, kann dann erschlossen werden, in welchem Raum sie sich befindet. Hieraus können dann Automationen programmiert werden, die z.B. dafür sorgen, dass das Licht in den Räumen angeht oder andere Vorgänge initialisiert werden. [83]

2.1.5 Presence Beacon – iBeacon

Abbildung 17: iBeacon [84]

Hierbei handelt es sich um Bluetooth Low Energy Beacons, die mit herumgetragen werden können und die Position des Besitzers anzeigen. Der Beacon bewirbt ständig seine Position mithilfe des Low Energy Bluetooth Protokolls. Man kann das Gerät mit einer App steuern, oder wie in diesem Fall, mithilfe eines Detektoren die Anwesenheitsnachrichten weiterleiten. Eine der beiden Seiten des Beacons ist ein Knopf, den man betätigen muss, um ihn anzuschalten. Er läuft mit einer Knopfzelle einen Monat lang bei stätigem Betrieb. Anschließend kann das Gerät geöffnet werden, um die Batterie auszuwechseln. [84]

2.1.6 HomeMatic Zentrale CCU2

Abbildung 18: HomeMatic CCU2 [85]

Die CCU2 (Central Computing Unit) ist das Herzstück eines HomeMatic Netzwerks. Das Gerät empfängt und sendet BidCoS-Nachrichten von und zu den Sensoren und Aktoren des Netzwerks. Mittels einer Frontend-Bedienoberfläche, die von jedem Browser zugänglich ist, wird dem Benutzer ermöglicht seine HomeMatic-Geräte fernzusteuern und zu automatisieren. [85] Verschiedene Strukturierungsoptionen bieten hier eine genaue Kontrolle der verschiedenen Funkmodule.

Die Zentrale verfügt über einen Ethernet-, einen MiniUSB- und einen normalen USB-Anschluss zur Datenübertragung und Netzwerkanbindung und außerdem einen Netzteileingang für die Stromversorgung. [86] Darüber hinaus ist es möglich eine SD-Karte einzustecken, um zusätzliche Programmierungen aufzuspielen.

Da der HomeMatic Hersteller eq3 sich bereit erklärt hat, seine ursprünglich proprietäre Software, unter der RAND-Z Lizenz zu veröffentlichen und Nutzern zu ermöglichen, diese für Entwicklungszwecke zu benutzen [87], ist es inzwischen auch möglich eine CCU2 auf einem Raspberry Pi nachzubauen. [88] Dazu ist nicht mehr nötig als eine maßgeschneiderte Raspbian Version auf das Gerät zu spielen. Somit kann man heutzutage bei HomeMatic auch schon von Open Source reden.

2.1.7 HomeMatic Dimmer & Rollladensteuerung

Da der CCU2 nur wenig Nutzen hat, ohne Geräte mit denen er kommunizieren kann, wurden noch zwei Aktoren von HomeMatic angeschafft, die sich in einem üblichen Smart Home Umfeld befinden könnten.

Abbildung 19: HM-LC-Dim1T-FM [89]

Hierbei handelt es sich um einen Unterputz-Dimmaktor, der das Dimmen von verschiedenen Glühlampen ermöglicht und darüber hinaus einige Konfigurationsmöglichkeiten bietet, wie die Rampenzeiten der Anschaltvorgänge, Zeitprofile oder die maximale Einschaltdauer. Der Dimmer hat einen eigenen Stromanschluss, somit ist er nicht auf die Lebenszeit von Batterien angewiesen. [89]

Abbildung 20: HM-LC-Bl1-FM [90]

Dieser Unterputz-Rollladenaktor ermöglicht eine ähnliche Funktionalität wie der Dimmer, seine Elektronik ist aber auf die Steuerung von verschiedenen Rollladenmotoren zugeschnitten. Auch dieses Gerät bietet die Möglichkeit die Steuerung auf die Gegebenheiten des Umfelds anzupassen und bezieht seine Leistung direkt vom Stromnetz. [90]

Beide Bauteile ermöglichen den Anschluss von Schaltern zur händischen Steuerung neben der Funkfernsteuerung und verfügen über eine LED zur Anzeige von Signalempfangsbestätigungen.

Abbildung 21: Der HomeMatic Aufbau mit Glühlampen

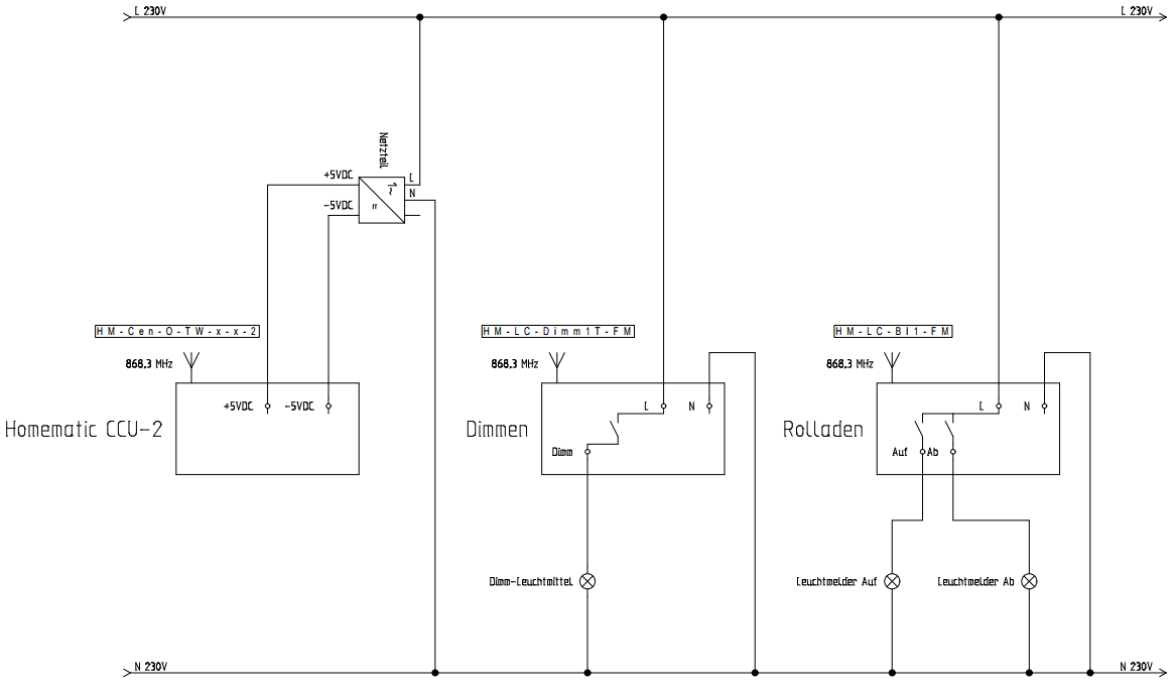

Abbildung 22: Der HomeMatic Aufbau als Schaltplan

Da für die Demonstration der Rollladenfunktionalität nicht notwendigerweise ein Rollladen benutzt werden muss und die Anschaffung eines solchen ein relativ großen Aufwand darstellt, wird für die Umsetzung dieses Beispielprojekts auf die Verwendung von zwei Lampen zurückgegriffen, um den Auf- und Abwärtsmechanismus zu simulieren. Somit wurden zusätzlich zu den HomeMatic Bauteilen noch drei Hallogenglühlampen und die notwendigen Kleinteile wie Fassungen, Leitungen und Klemmen eingekauft.

Sobald der Stromstecker eingesteckt wird, werden die Aktoren angeschaltet und sind nach wenigen Sekunden mit dem CCU2 steuerbar.

Weiterhin wurden einige Bauteile verwendet, auf die nicht näher eingegangen werden muss, wie USB-Kabel, USB-Bridges, SD-Karten oder Ethernet Kabel.

2.2 Realisierung auf verschiedenen Plattformen

In den folgenden Kapiteln werden die verschiedenen Prozesse erklärt, wie das beschriebene exemplarische Smart Home Szenario auf den unterschiedlichen Plattformen umgesetzt wird. Zwar hat die Realisierung in der Praxis natürlich sehr viele Anläufe und sehr viel Ausprobieren benötigt, dies wird aber größtenteils übersprungen.

Um dem Charakter eines Vergleichs gerecht zu werden, müssen jedoch zunächst Kriterien ausgewählt werden, anhand derer die Plattformen bewertet werden können, sodass später ein gerechter und sinnvoller Vergleich möglich ist. Die Kriterien sollen aufzeigen, wie gut sich die Plattformen für die Benutzung aus Sicht eines Entwicklers eignen. Je nach Kriterium handelt es sich um mehr oder weniger objektive Dinge, die zum Teil nur sehr schwierig zu quantifizieren sind. Im Folgenden also die Bewertungskriterien, die im nächsten Kapitel Grundlage des allgemeinen Vergleichs und der Bewertung sein sollen:

Installationsaufwand – Hiermit sind zwei Aspekte gemeint: Erstens, der Aufwand, der mit der reinen Plattforminstallation einhergeht, also die Arbeit, die aufgebracht werden muss, um von einem leeren Raspberry Pi zur laufenden Plattform zu gelangen. Und zweitens, die Installation der verwendeten Technologien, also wie viel Arbeit aufgebracht werden muss, um die Sensoren und Aktoren unter der Plattform zu installieren. Mit Aufwand sind sowohl reines Coding bzw. Scripting, als auch erfahrene Probleme und die Dauer der Installation gemeint.

Support – Keine (Open Source) Software funktioniert auf Anhieb ohne jegliche Probleme. In solchen Fällen ist das Vorhandensein von Ressourcen wie Wikis, Foren oder Support Chats im Internet ein sehr großer Vorteil. Dieses Kriterium bewertet wie hilfreich diese Ressourcen der jeweiligen Plattform sind.

User Experience – Hiermit sind sehr subjektive Eindrücke wie das Aussehen des Interfaces, die Nutzerführung oder die wahrgenommene Schnelligkeit der Plattform gemeint.

2.2.1 Werkzeuge

Da die Umsetzung in allen drei Fällen auf einem Raspberry Pi mit einem linuxbasierten Betriebssystem stattfindet, ergeben sich einige nützliche Werkzeuge, die die Installation, Wartung und Bedienung der Plattform vereinfachen und die projektunabhängig verwendet werden können.

2.2.1.1 SSH

SSH (Secure Shell) ist ein Netzwerkprotokoll, das im Juli 1995 erstmals entwickelt wurde. Sie wurde vom finnischen Informatiker Tatu Ylönen geschrieben, um Telnet und die Remote Shell rsh zu ersetzen. Inzwischen ist SSH besonders in Unix-Umgebungen zum Remote-Access Standard geworden und allgemein fast allgegenwärtig. Der große Vorteil von SSH gegenüber Protokollen wie Telnet ist, dass er ohne Mehraufwand genau dasselbe bietet und darüber hinaus noch mehr. [91]

Telnet, was zur Kommunikation mit einem entfernten Gerät genutzt wird, hat den großen Nachteil, dass es seine Daten in Klartext übermittelt, dazu gehören auch die Anmeldedaten des Benutzers. Das war zur Zeit seiner Erfindung, 1983, zwar noch kein Problem, weil Netzwerksicherheit noch kein großer Begriff war. [92] Heute jedoch wäre eine Übermittlung von Daten in Klartext ein extrem großes Sicherheitsrisiko, da ein Angreifer kein Problem hätte, die Daten abzuzweigen und dann auszulesen. Dieses Problem löst SSH: Alle Daten werden für den Benutzer transparent beim »Betreten« einer SSH-Verbindung verschlüsselt und ebenso transparent beim Verlassen der Verbindung auf dem entfernten Rechner wieder entschlüsselt. Somit gewährleistet SSH für den Benutzer Geheimhaltung durch die Verschlüsselung, Integrität der Daten durch sehr schwer zu fälschende Prüfsummen und Authentifizierung der beteiligten Parteien, da der Benutzer sich gegenüber dem Server identifizieren muss und sowohl Client als auch Server die gegenseitige Identifizierung zweifelsfrei feststellen müssen, bevor ein einziges Byte über die Verbindung fließt. [91]

Im Gegensatz zu Windows Systemen verfügen Unix Umgebungen über einen standardmäßig installierten SSH-Client, einfach ssh genannt. Mit diesem kann der Nutzer ganz leicht in der betriebssystemeigenen Shell eine Remote-Verbindung mit einem beliebigen Rechner starten, dieser muss jedoch einen SSH-Server installiert haben. Die Syntax lautet

```
ssh hostname [-l benutzername] [-p port-nummer]
```

Oder alternativ:

```
ssh [benutzername@]hostname [-p port-nummer]
```

Unabhängig davon, welche Variante man benutzt, sollte dieser Befehl eine Passworteingabe prompten. [93] Mac-Systeme, welche auch auf Unix basieren, besitzen dieselbe Funktion. Hierfür muss das Programm „Terminal“ geöffnet werden, was das Gegenstück zur Unix-shell ist. Es existieren verschiedene Arten von shells, Apple verwendet eine, die Bash genannt wird. [94]

2.2.1.2 PuTTY

Für die Bedienung eines Raspberry Pis gibt es zwei relevante Herangehensweisen: Entweder man nutzt ihn wie einen normalen Computer und schließt Maus und Tastatur per USB und den Monitor per HDMI an, oder man steuert ihn von einer anderen Maschine aus, was den Vorteil hat, dass die andere Maschine meist leistungsstärker ist und man seine Programmierarbeit auf einem Computer zentralisieren kann. Um den Raspberry Pi zu steuern wird das Kommunikationsprotokoll SSH verwendet. Da Windows-Maschinen standardmäßig nicht über einen SSH-Client verfügen, muss eine externe Applikation installiert werden. Hierfür gibt es verschiedene Alternativen: Command Line Programme wie Cygwin oder aber auch GUI basierte Programme wie WinSCP. An vielen Stellen wird aber empfohlen auf den SSH-Client PuTTY zurück zu greifen, da dieser weitaus kleiner in der Datengröße ist und auch seine Installation weitaus weniger kompliziert ist. PuTTY, in seiner reinen SSH/Telnet Ausgabe, hat nach dem Download eine Größe von 696 kb und benötigt keine herkömmliche Installation. Das Ausführen der .exe genügt und der Client ist bereit genutzt zu werden. [95]

PuTTY verfügt neben reiner SSH-Funktionalität noch über einen Telnet- und einen Rlogin-Client und ermöglicht unter anderem auch die serielle Verbindung mit Geräten. Mehrere Einstellungen ermöglichen dem Nutzer seine Verwendung des Programms zu optimieren, so z.B. die Anpassung des Terminalfensters, das Speichern von Sitzungen oder das Loggen der Standardausgabe, die vom Remote-Server geschickt wird. [91]

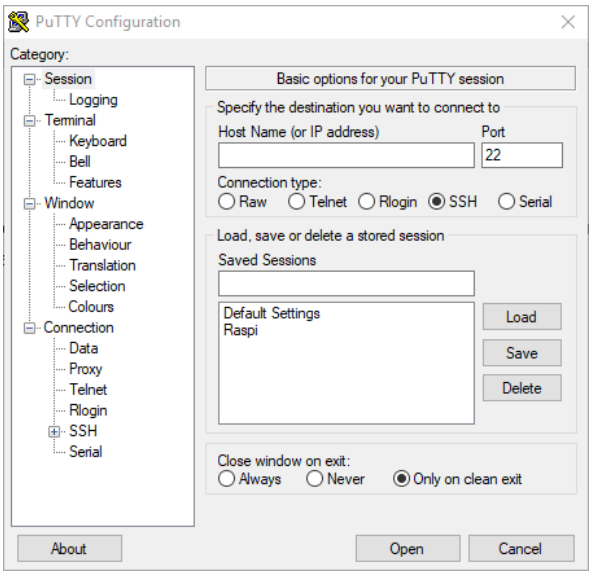

Abbildung 23: Das Programm PuTTY

2.2.1.3 Samba

Zwar ist es durchaus möglich alle benötigten Schritte und Aktionen in PuTTY via SSH auszuführen, jedoch ist es oftmals einfacher mit einem modernerem Interface zu arbeiten. Besonders das Bearbeiten von Systemdateien wird durch die Verwendung von hochwertigen Editierprogrammen erleichtert. Dies wird ermöglicht, indem das Dateisystem des Raspberry Pis im Netzwerk verfügbar gemacht wird. Daraufhin kann man von jedem Computer im Netzwerk auf die Dateien zugreifen und sie bearbeiten. Hierfür wird das Netzwerkfreigabeprogramm „Samba“ installiert.

Nachdem das Programm auf dem Raspberry Pi mit Hilfe des APT installiert wurde, muss noch die Konfigurationsdatei unter `/etc/samba/samba.conf` bearbeitet werden, indem der Inhalt mit den folgenden Einstellungen ersetzt wird:

```
[global]
netbios name = RP2
server string = The Pi File Center
workgroup = WORKGROUP
hosts allow =
socket options = TCP_NODELAY IPTOS_LOWDELAY SO_RCVBUF=65536 SO_SNDBUF=65536
remote announce =
remote browse sync =

[HOMEPI]
path = /
comment = No comment
browsable = yes
read only = no
valid users =
writable = yes
guest ok = yes
public = yes
create mask = 0777
directory mask = 0777
force user = root
force create mode = 0777
force directory mode = 0777
hosts allow =
```

Anschließend muss noch ein Samba Nutzer mit dem `smbpasswd` angelegt werden und schon kann von jedem Computer im Netzwerk auf das Raspberry Pi Dateisystem zugegriffen werden.

2.2.1.4 Mosquitto & HappyBubbles Presence Software

Die MQTT Funktionalität wird mit dem Programm *Mosquitto* gewährleistet. Genauer gesagt wird der Raspberry Pi als Broker eingestellt, sodass er die MQTT Nachrichten empfangen und verteilen kann. Mosquitto kann einfach mit dem APT installiert werden. Anschließend kann die Funktionalität getestet werden, indem zwei SSH Verbindungen geöffnet werden und in der einen auf einen Kanal abonniert wird und in der anderen eine Nachricht in jenen Kanal veröffentlicht wird, welche dann wiederum in der anderen Verbindung angezeigt werden sollte. Die Befehle dafür lauten:

```
mosquito_sub -d -t “test”
```

```
mosquito_pub -d -t “test” -m “Hallo Welt!“
```

“-d” aktiviert den Entwicklungsmodus, in welchem zu jeder Nachricht mehr Informationen angezeigt werden, „-t“ spezifiziert das Thema und „-m“ die Nachricht selbst. [96] Mosquitto verfügt auch über die Möglichkeit eine Nutzername/Passwort Kombination einzustellen, welche es verhindert, dass nicht autorisierte Benutzer Nachrichten an den Broker schicken. Für das Projekt war dies aber nicht notwendig. Nachdem Mosquitto installiert und getestet wurde, kann der Pi als Broker verwendet werden.

Damit die Präsenzerkennung richtig funktioniert, muss die HappyBubbles Technologie konfiguriert werden. Der erste Schritt ist es das Gerät via MicroUSB mit Strom zu versorgen. Violettes Blinken der LED signalisiert, dass es angeschaltet ist. Anschließend drückt man lang auf den *Config* Knopf, bis die LED orange leuchtet. Jetzt befindet sich der Detektor im Konfigurationsmodus und hat ein WLAN-Netzwerk namens „happy_bubbles_ble“ erstellt, in das man sich einwählen kann. Von hier aus steuert man in einem Browser die Adresse 192.168.4.1 an und gelangt ins HappyBubbles Frontend, von wo aus man die WLAN- und MQTT-Einstellungen konfigurieren kann. Im Reiter WiFi-Setup muss das Heimnetzwerk ausgewählt werden und das Passwort eingetragen werden. Nach einem Klick auf „Connect!“, sollte im Reiter „Home“ der WiFi Status als „got IP address“ angezeigt werden. Im Reiter MQTT Setup müssen noch die Daten des Brokers eingetragen werden, also IP-Adresse, Port, Benutzername und Passwort. Anschließend sollte wiederum im Reiter „Home“ der MQTT Status als „enabled/connected“ angezeigt werden.

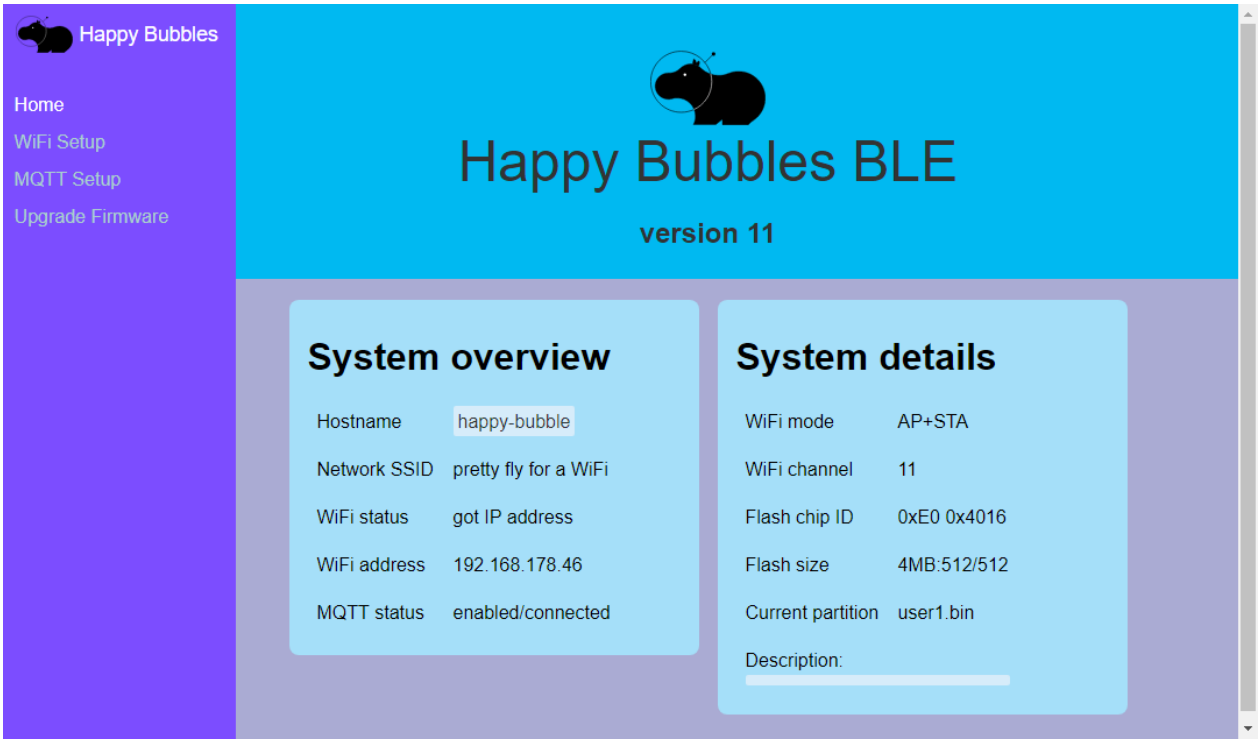

Abbildung 24: HappyBubbles Frontend

Nun kann der Detektor aus dem Konfigurationsmodus genommen werden, indem wieder lang auf den *Config* Knopf gedrückt wird. Ein erneutes violettes Blinken signalisiert, dass das Gerät sich nun im normalen Modus befindet. Um nun zu testen, ob die Präsenzerkennung richtig funktioniert und der Detektor seine Nachrichten per MQTT absendet, kann im PuTTY Fenster der MQTT-Verkehr abgehört werden, der über den Broker vermittelt wird. Hierfür können zunächst alle Channels abgehört werden:

```
mosquitto_sub -d -u username -P password
-t “#”
```

In einem durchschnittlichen Haushalt kann der Traffic hier jedoch sehr hoch sein, da einige Geräte Bluetooth Signale versenden. HappyBubbles leitet grundsätzlich alle Bluetooth Advertisements an den

Mosquitto ermöglicht das sogenannte wildcarden für das MQTT-Topic. Damit ist gemeint, dass man nicht spezifiziert, genau welchen Channel man abonniert, sondern anstelle des Namens des Channels ein Rautenzeichen setzt (#). So bekommt man die Veröffentlichungen aus allen Channels. Besonders nützlich ist dieses Feature, wenn man nicht genau weiß, zu welchem Channel Nachrichten veröffentlicht werden.

Außerdem ist es möglich Subchannels zu wildcarden, um alle Veröffentlichungen aus den Subchannels zu bekommen („channel/subchannel/#“).

Channel `happy-bubbles/ble/<the-hostname>/raw/<the-bluetooth-MAC-address>` in JSON Format weiter. Ein solcher Eintrag könnte so aussehen:

```
Client mosqsub/3623-raspberryp received PUBLISH (d0, q0, r0, m0, 'happy-
bubbles/ble/happy-bubble/raw/000d44f5701a', ... (173 bytes))
{"hostname": "happy-bubble",
"mac": "000d44f5701a",
"rssi": -87,
"is_scan_response": "0",
"type": "0",
"data": "02010602080003036 1fe10ff030000072f00000570700d6e9f4e12"}
```

Handelt es sich bei dem werbenden Gerät um einen iBeacon, wird eine ähnliche JSON-Nachricht im Channel `happy-bubbles/ble/<the-hostname>/ibeacon/<the-iBeacon-UUID>` veröffentlicht. [81] Natürlich muss hierfür der iBeacon aktiviert werden. Nachdem der Detektor nun die Anwesenheit der Beacons korrekt berichtet, kann die Einbindung in Home-Assistant beginnen. Hierfür stellt HappyBubbles die *Presence* Software bereit. Im PuTTY Fenster muss zunächst der folgende Befehl ausgeführt werden, mit dem das Installationsskript heruntergeladen wird:

```
wget https://github.com/happy-bubbles/presence/releases/download/1.6.2/presence_rpi3_install.sh
```

Anschließend wird das Skript ausgeführt:

```
bash presence_rpi3_install.sh
```

Jetzt kann unter der Adresse http://raspberrypi:5555/ im Browser das Interface der Presence Software erreicht werden. Unter „Latest Beacons Seen“ werden alle Beacons in der Nähe aufgelistet. Diese Auflistung korrespondiert mit der Ausgabe aller Channel Aktivitäten (Wildcard #). Beim Eintrag mit dem Wert „iBeacon“ bei „Beacon Type“ ist der verwendete iBeacon. Mit „Add this beacon“ kann er gespeichert werden, woraufhin er unter der Home Ansicht permanent angezeigt wird. Sobald der Beacon gespeichert wurde, wird er und seine Distanz zum Detektor im Kanal „happy-bubbles/presence/ha/happy-bubble“ periodisch veröffentlicht.

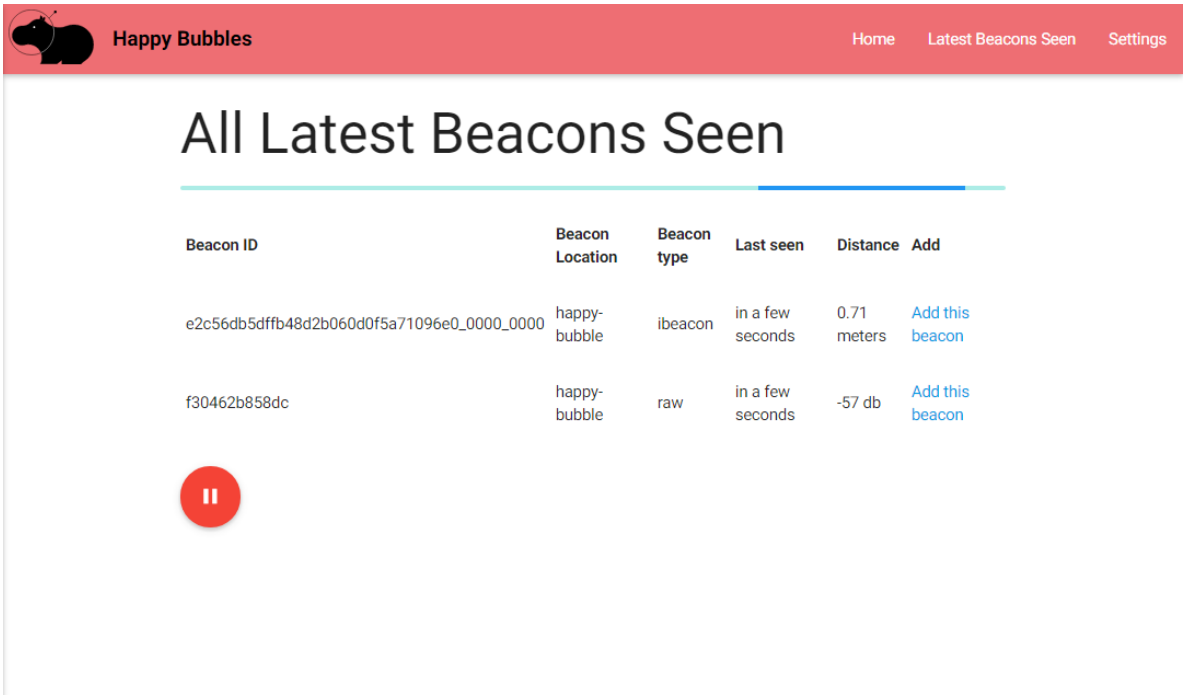

Abbildung 25: Die Benutzeroberfläche der Presence Software

2.2.2 Home-Assistant

2.2.2.1 Installation

Home-Assistant ist für den Raspberry Pi optimiert und bietet hierfür auch die einfachsten und robustesten Software-Lösungen. Beispielsweise existiert der Raspberry Pi All-In-One-Installer – ein Skript, das automatisch einen vollständigen Home-Assistant Server installiert, inklusive MQTT und Z-Wave Support. Einzige Voraussetzung ist eine frische Installation von Raspbian und SSH Zugang, wofür wiederum eine Netzwerkverbindung notwendig ist. [97] Natürlich ist es auch möglich mit Monitor, Tastatur und Maus den Raspberry Pi zu bedienen, dagegen stellt SSH jedoch die einfachere Variante dar. Alternativ kann Home-Assistant auch manuell installiert werden. Hierfür muss eine Python Virtual Environment eingerichtet werden und anschließend das *homeassistant* Python Paket installiert werden. [98]

Eine andere Art und Weise Home-Assistant auf dem Raspberry Pi einzurichten ist Hassbian, eine maßgeschneiderte Variante von Raspbian, welches laut Webseite die einfachste Möglichkeit ist, Home-Assistant zu installieren. Hierfür muss nur die Hassbian Image auf die SD Karte geflasht werden, wie es auch bei Raspbian üblich ist. Danach ist Home-Assistant schon betriebstauglich. Weitere Werkzeuge, die in Hassbian eingebaut sind, machen die Nutzung des Systems einfacher als mit der All-In-One Installation oder gar einer manuellen Installation. [99]

Seit Mai 2017 existiert für Home-Assistant außerdem eine weitere Art das System zu installieren: Hass.io. Diese Installation verfügt über den sogenannten Supervisor, der es Hass.io erlaubt auf Systemfunktionen zuzugreifen, wodurch die Bedienung von Home-Assistant weitaus einfacher wird, da Kommandozeilenbefehle überhaupt nicht mehr notwendig sind. Außerdem vereinfacht Hass.io das Installieren von „Add-On“, also Programmen wie Samba oder Mosquitto (Vgl. 2.2.1), welche notwendig für einen effizienten Workflow sind. [100]

Für dieses Projekt wurde auf die Installation mit dem AIO-Installer zurückgegriffen, welcher als Shell Skript sehr transparent ist und im Grunde auch Schritt für Schritt nachgemacht werden kann, was einer manuellen Installation gleicht. Hassbian ist zwar eventuell die einfachere Lösung, jedoch wird der Vergleich zu den anderen Plattformen einfacher, wenn bei allen eine ungefähr ähnliche Installationsvariante gewählt wird.

2.2.2.2 *Oberfläche*

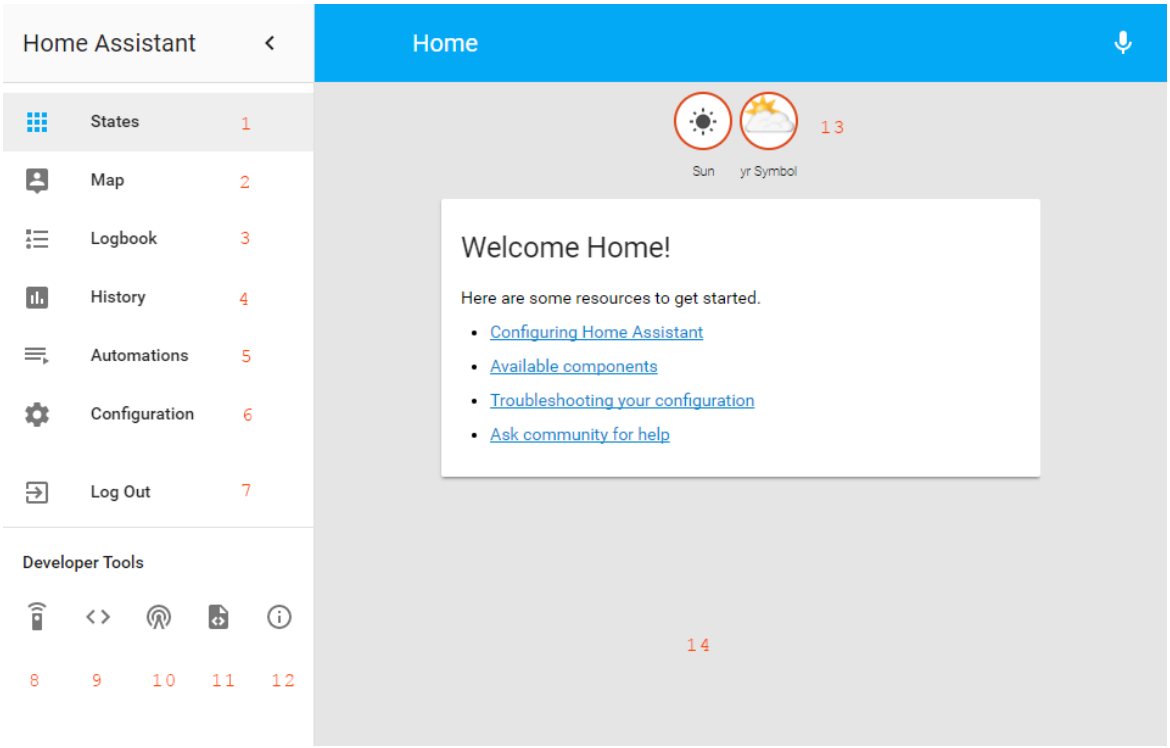

Abbildung 26: Die Home-Assistant Benutzeroberfläche

Die Home-Assistant Oberfläche kann nach der Installation unter der URL `<Raspberry Pi IP>:8123` aufgerufen werden. Zu Beginn ist sie noch leer, da die Komponenten zunächst implementiert werden müssen.

Auf der Startoberfläche befinden sich die folgenden Elemente:

1) Der Link zur Startoberfläche selbst
2) Eine Karte, die alle definierten Zonen anzeigt [101]
3) Wie der Name schon suggeriert, ein Logbuch, das Aktivitäten protokolliert, jedoch nur „große“ Aktivitäten
4) Eine Visualisierung der Statusveränderung der einzelnen Geräte
5) Das Menü zur Erstellung von Automationen
6) Verschiedene Einstellungen wie Konfigurierungsüberprüfung, Server Neustart u.A.
7) Log Out – Dieser bewirkt aber nur wirklich etwas, wenn ein Passwort eingestellt ist
8) Services – Hier können Services von der Oberfläche aus ausgelöst werden
9) States – Eine Auflistung aller Geräte inklusive ihrer Status und Attribute
10) Events – Von hier aus können Events ausgelöst werden, wie ein Server Stopp

11) Templates
12) About – Grundlegende Informationen über den Server und ein Fehlerlog
13) Die Bubbles, welche einzelne Entities repräsentieren
14) Der Hauptbereich, wo Cards angelegt werden können und Geräte angezeigt werden

2.2.2.3 Konfiguration und Betrieb

In Home-Assistant findet grundsätzlich jede Art von Konfiguration in der dafür zuständigen YAML Datei statt. Diese heißt `configuration.yaml` und befindet sich in `/home/pi/homeassistant/.homeassistant/`. Dieser Pfad beinhaltet grundsätzlich alle Konfigurationsdateien und wird allgemein als „Home"-Pfad verwendet.

Zu Beginn sind schon einige Einstellungen vorgegeben, die den Einstieg erleichtern. Unter anderem die Folgenden:

- *discovery:* Ermöglicht das automatische Erkennen von Geräten im Netzwerk, die in Home-Assistant eingebunden werden können. Ein Google Chromecast würde beispielsweise sofort erkannt und eingebunden werden, so auch viele TVs oder ähnliche Geräte
- *homeasstistant:* Diese Liste beinhaltet Einstellungen, wie den Namen des Systems oder die Koordinaten des Hauses, welche genutzt werden, um die Uhrzeit und den Sonnenstand zu berechnen.
- *logger:* Diese Option ist zwar eingangs noch deaktiviert, jedoch kann sie durch das Hinzufügen des Parameters *default: info* so aktiviert werden, dass in einer Datei alle Systemausgaben gespeichert werden (*home-assistant.log*)

Wie grundlegende Systemeinstellungen, müssen auch alle zusätzlichen Technologien, die man einbinden möchte, in der `configuration.yaml` angelegt werden.

2.2.2.4 Automationen

Automationen sind festgelegte Abläufe, die das System befolgt und bestehen aus drei Teilen: Trigger (Auslöser), Condition (Bedingung) und Action (Aktion). Folgendes Beispiel soll die Funktionsweise von Automationen verbildlichen: [102]

```
(trigger)      Wenn Paul nach Hause kommt
(condition)    und es ist nach Sonnenuntergang:
(action)       Schalte die Lichter im Wohnzimmer an
```

- Die erste Zeile ist der Trigger der Automation. Sie beschreiben die Ereignisse, die eintreten müssen, um eine Automation auszulösen. In diesem Fall ist es das Heimkommen einer Person, welches in Home-Assistant anhand der Statusänderung eines Sensors beobachtet werden kann.
- Die zweite Zeile ist die Condition. Conditions sind optionale Überprüfungen, welche die Automation auf eine ausgewählte Zahl von Anwendungsfällen limitieren können. Wenn die Bedingung nicht erfüllt wird, wird die Automation nicht ausgeführt. In diesem Fall wird geprüft, ob die Sonne schon untergegangen ist. Es kann jedoch auf eine Vielzahl von Zuständen geprüft werden wie Zeit, Anwesenheit, Sensorwerte usw.
- Der dritte Teil ist die Aktion, die ausgeführt wird, wenn die Automation anhand des Triggers ausgelöst wird und die Bedingungen erfüllt sind. In diesem Beispiel werden die Lichter im Wohnzimmer angeschaltet, es könnte aber auch die Temperatur des Thermostats geändert werden, eine Mail geschickt werden uvm.

Im Gegensatz zu anderen Systemen, ist es in Home-Assistant nicht vorgesehen, dass Skripte geschrieben werden, vielmehr handelt es sich hier um feste Blöcke als klassische Abläufe. Insofern ist es komplizierter oder zumindest aufwendiger, komplizierte Szenarien umzusetzen, wie eine einfache „if else“ Bedingung. Zwar sind Bedingungen Teil des Automationsmodells, jedoch gibt es keine Möglichkeit zu spezifizieren, was passieren soll, wenn die Bedingung nicht eintritt. Die einfachste, obgleich sehr unelegante Herangehensweise ist es, für den selben Auslöser zwei verschiedene Automationsblöcke zu schreiben, die unterschiedliche Bedingungen haben.

2.2.2.5 Z-Wave

Um Z-Wave in Home-Assistant einzubinden, muss der folgende Eintrag in die `configuration.yaml` gemacht werden:

```
zwave:
  usb_path: /dev/ttyUSB0
```

Natürlich vorausgesetzt, der USB Pfad lautet `/dev/ttyUSB0`. Anschließend muss der MultiSensor integriert werden. Zwar verfügt Home-Assistant inzwischen über ein eigenes Z-Wave Menü, dieses ist aber nicht sehr zuverlässig, weshalb hier auf das Open ZWave Control Panel zurückgegriffen wird. Dieses wird bei Home-Assistant mitinstalliert, unabhängig davon, welchen Installationsweg man verwendet hat. Bevor es gestartet werden kann, muss zunächst der Home-Assistant Server gestoppt werden (*Configuration* Menü), anschließend wird OZCWP gestartet:

```
cd /srv/homeassistant/src/open-zwave-control-panel/ && sudo ./ozwcp 8888
```

Im Control Panel kann Z-Wave sehr einfach gesteuert werden, Einstellungen geändert werden, Nodes hinzugefügt oder entfernt werden uvm. Zunächst muss noch der USB-Pfad des Controllers eingegeben werden, anschließend wird das komplette Z-Wave Netzwerk angezeigt. Um nun den MultiSensor hinzuzufügen wird unter *Controller* im Dropdown die Funktion *Add Node* ausgewählt und auf *Go* geklickt. Anschließend muss der MultiSensor betätigt werden, danach sollte er im Control Panel angezeigt werden.

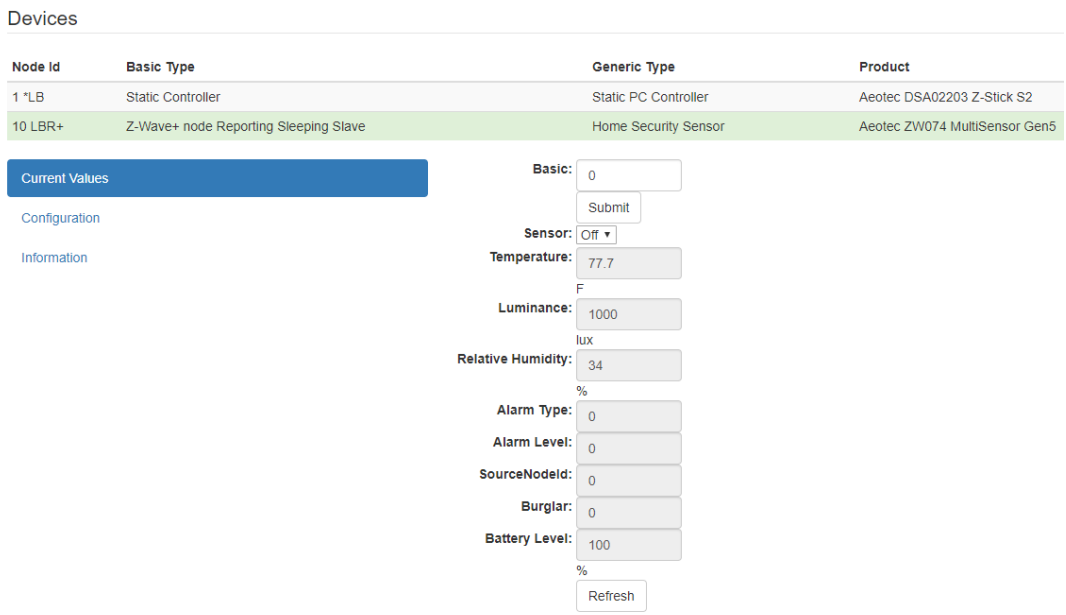

Abbildung 27: Das Open Zwave Control Panel

Um ihn nun für das Projekt betriebsfähig zu machen, müssen noch einige Einstellungen geändert werden. Im Menü *Configuration* werden die folgenden Einstellungen übernommen:

```
Enable Motion Sensor: Enabled
Command Options: Binary Sensor Report
On time: 10
Group 1 Reports: 224
```

Außerdem kann bei Bedarf die Frequenz der Updates der anderen Sensoren geändert werden - `Group 1 Interval`. Im Werkszustand liegt diese

Z-Wave Parameter sind 8 Byte groß. Das Bitset von Group 1 Reports *lautet:*
0 – Batteriestand
5 – Temperatur
6 – Luftfeuchtigkeit
7 – Lichtstärke
Wenn der Sensor also alle Werte senden soll, außer den Batteriestand (USB-Modus), lautet die Rechnung wie folgt:

$$0 * (2^0) + 2^5 + 2^6 + 2^7 = 224$$

bei 30 Minuten. Anschließend kann das Control Panel gestoppt (in PuTTY `Strg+C`) und der Home-Assistant Server wieder gestartet werden.

Jetzt sollte der Bewegungssensor wie gewünscht funktionieren: Wenn er eine Bewegung registriert, ändert sich sein Status für 10 Sekunden in „on“ und wechselt danach wieder zu „off“. Außerdem wird er als „Bubble“ angezeigt.

Abbildung 28: Home-Assistant „Bubbles“ inklusive Bewegungsmelderanzeige

2.2.2.6 HomeMatic

Die Anbindung von HomeMatic in Home-Assistant ist sehr einfach. Der einzige notwendige Schritt ist das Einfügen der folgenden Zeilen in die `configuration.yaml` Datei:

```
homematic:
  hosts:
    wireless:
      ip: <CCU2 IP>
```

Vorausgesetzt die Geräte sind im CCU2 registriert, sollten sie sofort in der Home-Assistant Oberfläche als bedienbare Cards erscheinen.

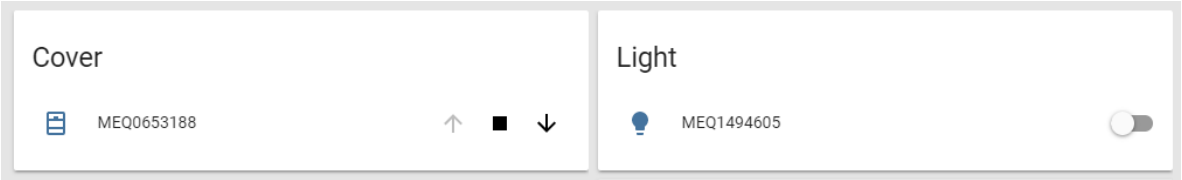

Abbildung 29: HomeMatic Gerätesteuerung in Home-Assistant

2.2.2.7 MQTT

Um MQTT in Home-Assistant zu aktivieren, muss die Komponente in die `configuration.yaml` Datei eingetragen werden, indem man den Broker spezifiziert. Die notwendigen Zeilen dafür lauten:

```
mqtt:
       broker: <raspberry pi IP>
```

Da die Mosquitto Broker Software auf dem Raspberry Pi läuft, muss dessen IP eingetragen werden. Anschließend sollte man den Server neustarten und kann danach die Funktionalität testen. Im Services Fenster, welches in den Developer Tools zu finden ist, können manuell Services ausgeführt werden, darunter auch MQTT Publishes. Für die *Domain* wird MQTT ausgewählt und für den *Service* Publish. Danach muss eine Veröffentlichung im JSON Format geschrieben werden:

```
{
       "topic": "test",
       "payload": "Hallo Welt!"
}
```

Vorausgesetzt in der SSH Verbindung wurde das Topic „Test“ abonniert, sollte die Nachricht dort ankommen, also funktioniert MQTT nun auch in Home-Assistant.

Anschließend muss der iBeacon in Home-Assistant als Sensor angemeldet werden. Dafür wird der folgende Eintrag in die configuration.yaml Datei eingefügt: [103]

```
sensor:
  - platform: mqtt_room
    device_id: e2c56db5dffb48d2b060d0f5a71096e0_0000_0000
    name: 'iBeacon'
    state_topic: 'happy-bubbles/presence/ha'
    away_timeout: 10
```

- *platform: mqtt_room* muss immer geschrieben werden.
- *device_id* ist die Beacon ID, die auch im Presence Frontend angezeigt wird
- *name* ist der Anzeigename, der im Home-Assistant Interface angezeigt wird

- *state_topic* muss immer auf ‚happy-bubbles/presence/ha' gesetzt sein, weil in diesem Channel die relevanten Nachrichten veröffentlicht werden
- *away_timeout* (in Sekunden) ist die Zeit, die gewartet wird, nachdem das erste Mal kein Advertisement vom Beacon empfangen wurde, um das Gerät als „away" anzuzeigen

Nach einem Neustart des Servers sollte nun in der oberen Sensorenleiste auch ein Eintrag für den iBeacon zu sehen sein, die angeklickt werden kann, um sowohl eine Zeitleiste des Zustands als auch die Distanz zum Detektor anzuzeigen. Aufgrund der Interaktion von Home-Assistant und der Presence Software, wird der Beacon als „happy-bubble" angezeigt, wenn er in Reichweite ist und als „away", wenn er länger als 10 Sekunden nicht in Reichweite war.

Abbildung 30: Sensorenleiste mit iBeacon

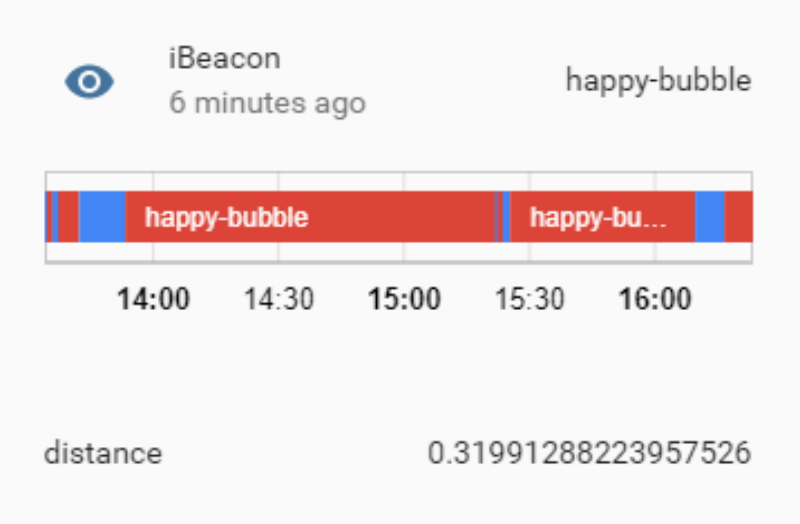

Abbildung 31: Detailansicht iBeacon

2.2.2.8 Umsetzung des Szenarios

Da beim vorliegenden Szenario, abhängig davon, ob der Benutzer daheim ist oder nicht, unterschiedliche Aktionen ausgeführt werden sollen, handelt es sich um eine „if else“ Bedingung, welche, wie schon angesprochen (vgl. 3.2.3.4), in Home-Assistant schwierig umzusetzen ist. Der einfachste Weg, um das gewünschte Verhalten zu erhalten, ist es zwei Automationsblöcke anzulegen. Zunächst aber muss erst mal eine „Dummy“-Automation angelegt werden, da Home-Assistant auch nach langen Versuchen nicht das selbstständige Anlegen von Automationen akzeptiert hat. Grund dafür ist, dass die Plattform seit Version 0.45 den sogenannten *Automation Editor* verwendet, ein Formular in der Benutzeroberfläche, die es dem Benutzer erlaubt, Automationen zu erstellen. [104, 105] Da sich mit der Einführung dieses Features jedoch die gesamte Automationsstruktur geändert hat, kommt es noch zu Fehlern. Das derzeitige Vorgehen beinhaltet also das Anlegen einer Automation im Editor und anschließend das Bearbeiten der Automation in einem externen Code Programm wie Notepad++. Besonders notwendig wird dieses Vorgehen dann, wenn Conditions verwendet werden sollen, da diese derzeit nicht vom Editor unterstützt werden.

Nachdem die Dummy-Automation nun mit Platzhalterwerten angelegt wurde, kann sie im Heim Verzeichnis des Raspberry Pis gefunden werden, im selben Verzeichnis, in der sich auch die `configuration.yaml` Datei befindet. Alle Automationen werden in der `automations.yaml` Datei angelegt. Ein weiterer Fehler des Editors ist, dass bei der „Übersetzung“ der Automationen die Struktur des Codes in falscher Reihenfolge angelegt wird. Glücklicherweise hat das keinerlei Auswirkung auf das Verhalten der Automation. Die Dummy Automation sieht dementsprechend so aus:

```
- action:
  - data:
      entity_id: licht
    service: licht an
  alias: “Dummy”
  id: '123123123'
  trigger:
  - entity_id: bewegungsmelder
    from: 'off'
    platform: state
    to: 'on'
```

Wie man erkennt, wird die *Action* zuerst angegeben, dann Alias und ID und dann erst der *Trigger*. Der Alias ist der *friendly name* der Automation und die ID ist eine einzigartige Zahl,

anhand derer die Automation identifiziert wird. Eine im Editor erstellte Automation hat immer eine zufällige ID. Was nun also noch fehlt sind die richtigen Namen der Services und Entitys und die *Condition*.

Fügt man diese Dinge ein, sieht die Automation folgendermaßen aus:

```
- action:
  - data:
      entity_id: light.meq1494605
    service: light.turn_on
  alias: Jemand ist daheim.
  id: '123123123'
  trigger:
  - entity_id: binary_sensor.aeotec_zw074_multisensor_gen5_sensor
    from: 'off '
    platform: state
    to: 'on'
  condition:
    condition: state
    entity_id: sensor.ibeacon
    state: 'happy-bubble'
```

Diese Automation schaltet das Licht an, unter der Bedingung, dass jemand daheim ist, also dass der Beacon den Zustand „happy-bubble“ besitzt (vgl. 3.2.3.7).

Was jetzt noch fehlt ist die „else“ Automation, also die Automation, die eine Mail abschickt, wenn der Bewegungsmelder ausgelöst wird, der Beacon aber den Zustand „away“ besitzt. Dafür muss zunächst noch die Mail Funktionalität integriert werden. Hierfür wird der folgende Eintrag in die `configuration.yaml` Datei eingefügt:

```
notify:
  - name: mail
    platform: smtp
    server: <smtp-server des sender-providers>
    port: 587
    timeout: 15
    sender: <emai-adresse des senders>
    starttls: true
```

```
    username: <username>
    password: <password>
    recipient:
      - <email-adresse des empfängers>
    sender_name: My Home Assistant
```

Daraufhin muss, um eine Mail abzuschicken, nur noch der Titel und der Körper der Nachricht angegeben werden. Die vollständige „else“ Automation sieht dementsprechend so aus:

```
- action:
  - data:
      title: "Intruder Alert!"
      message: "Niemand ist daheim, aber es hat sich was bewegt!"
    service: notify.mail
  alias: Niemand ist daheim.
  id: '234234234'
  trigger:
  - entity_id: binary_sensor.aeotec_zw074_multisensor_gen5_sensor
    from: 'off '
    platform: state
    to: 'on'
  condition:
    condition: state
    entity_id: sensor.ibeacon
    state: 'away'
```

Die Umsetzung mit Hilfe mehrerer Automationsblöcke ist zwar nicht die technisch korrekteste Lösung, jedoch funktioniert sie ohne weitere Plugins o.Ä. und lässt sich dadurch mit reiner Home-Assistant Kernfunktionalität umsetzen.

2.2.3 FHEM

2.2.3.1 Installation

Die Installation von FHEM ist mit den zwei herkömmlichen Installationswegen von Linux möglich. Entweder man nutzt das weit verbreitete `APT` und installiert das FHEM Paket, nachdem man das Repository zu seiner Source Liste hinzufügt hat, oder man benutzt die grundlegendere Methode mit `dpkg`. Hierfür müssen zunächst noch die vorausgesetzten Pakete installiert werden, daraufhin wird das Debian Paket manuell heruntergeladen und installiert und anschließend noch das FHEM Nutzerprofil angelegt. Beide Varianten führen zu demselben Resultat. [106] Unabhängig davon, welchen Weg man wählt, muss vorher schon eine frische Raspbian Installation vorliegen und es muss auf das System zugegriffen werden, mit SSH oder mit Monitor, Maus und Tastatur.

Die FHEM Webseite erwähnt den sogenannten „FHEM configurator“, der die Installation noch einfacher machen soll, dieser steht aber zur Zeit der Verfassung noch nicht zur Verfügung. [106] Eine Art von auf die Plattform angepasstes Raspbian existiert für FHEM leider nicht.

2.2.3.2 Oberfläche

Die Home-Assistant Oberfläche kann nach der Installation unter der URL `<Raspberry Pi IP>:8083` aufgerufen werden. Sie sieht folgendermaßen aus:

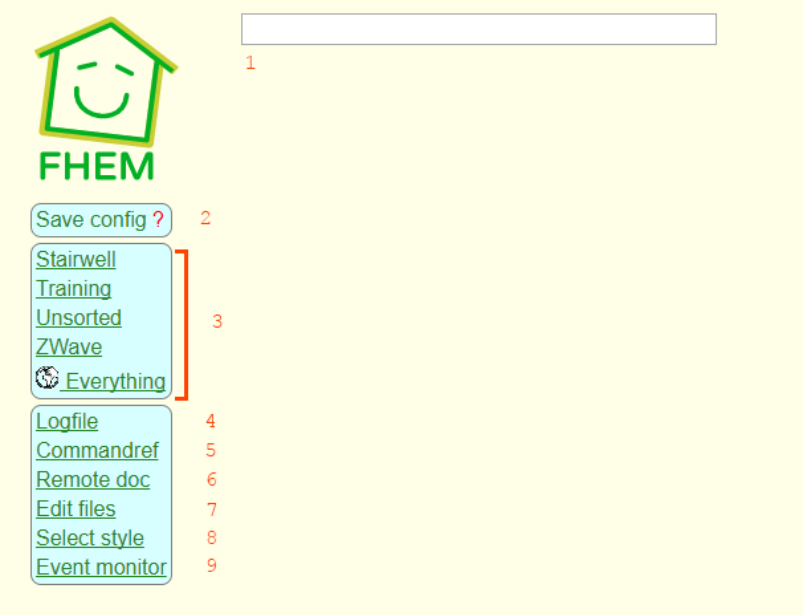

Abbildung 32: Die FHEM Benutzeroberfläche

1) Die Frontend Eingabemaske, wo Befehle eingegeben werden können. Befehle werden nur mit der Enter Taste abgesendet, es existiert kein „Absenden“ Button o.Ä.
2) Dieser Button muss betätigt werden, wenn Änderungen nach einem FHEM Neustart noch vorhanden sein sollen. Bis er betätigt wurde, also bis das rote Fragezeichen verschwindet, sind die Änderungen nur in einer temporären Datei gespeichert.
3) Die Device Räume. „Unsorted“ und „Everything“ sind immer da, die restlichen Räume wurden händisch angelegt.
4) Hier kann die Log Dateien im Browser geöffnet werden. Da sie aber gerne mehrere Kilobytes groß sein kann, wird davon abgeraten.
5) Hier findet sich eine ausführliche Offline Referenz aller Befehle.
6) Eine weitere Referenz, diese befindet sich jedoch Online auf der FHEM Webseite
7) Hier können verschiedene Dateien bearbeitet werden, ohne das Dateiverzeichnis öffnen zu müssen.
8) Auswahl des Anzeigestils. Rein optisch.
9) Der Event Monitor hat dieselbe Funktion wie die Log Datei, jedoch ist dieser „live“ und man bekommt nur die Einträge angezeigt, die seit dem Moment angelegt wurden, als man den Monitor gestartet hat. Es empfiehlt sich, den Event Monitor in einem parallelen Tab geöffnet zu haben.

2.2.3.3 *Konfiguration und Betrieb*

In FHEM werden alle Konfigurationsbefehle entweder in die Eingabemaske der Frontend Oberfläche eingegeben, oder manuell in die Konfigurationsdatei `fhem.cfg` im Verzeichnis `/opt/fhem/`. Die Eingaben, die man in die Maske macht, landen fast unverändert in der Datei. Es muss jedoch auf den Button *save config* gedrückt werden, damit die Änderungen, die man in der Maske abgeschickt hat, auch nach einem FHEM Neustart noch vorhanden sind.

Vorteil der direkten Eingabe im Interface ist der direkte Feedback vom System, da man nach der Eingabe eine Rückmeldung bekommt. Nachteil ist die schlichtweg unvorteilhafte User Experience. Da es sich um ein normales HTML `input` Element des Typs `text` handelt, ist die Eingabe nur eine Zeile breit und zeigt bei Fokussierung die vergangenen Eingaben an, was schnell irritierend wirken kann. Darüber hinaus ist keinerlei Highlighting oder Autokorrektur des Codes gegeben.

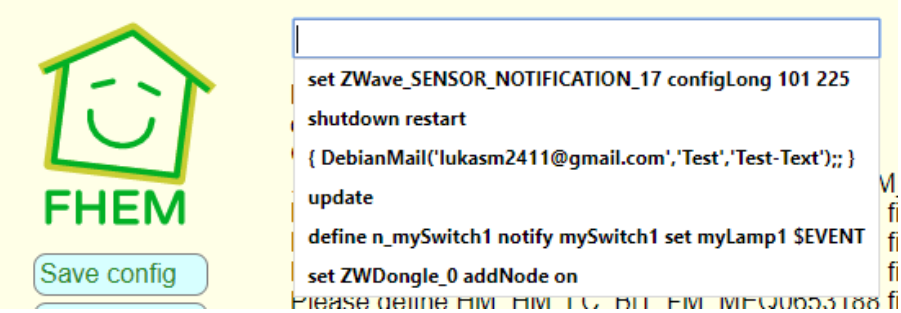

Abbildung 33: Die Eingabemaske mit HTML-üblicher Eingabehistorie

Das Grundgerüst von FHEM sind "Devices", also Geräte. Grundsätzlich sind alle Komponenten, die in FHEM eingebaut sind Devices. Dazu gehört beispielsweise auch die Benutzeroberfläche (*FHEMWEB*), ebenso gibt es Devices ohne physische Geräte, die dazu benutzt werden können, um eine Variable zu speichern. Um ein solches Device anzulegen, wird der folgende Befehl benutzt:

```
define <device-name> <device-typ>
```

Der Device Name ist frei wählbar, wobei er einzigartig sein muss – keine zwei Devices dürfen denselben Namen tragen. Jedes Device hat genau einen "Typ". Dieser Typ legt gleichzeitig die Perl-Befehlsdatei (man spricht präziser vom "Modul") fest, in der bestimmte Routinen und Eigenschaften festgehalten sind. Ist etwa ein Device vom Typ dummy, so weiß FHEM, dass die Routinen und Eigenschaften im Modul 98_dummy.pm liegen. (Die Zahl 98

hatte früher eine Bedeutung, heute ist das nicht mehr der Fall, aus historischen Gründen wird sie aber immer noch verwendet.) [107]

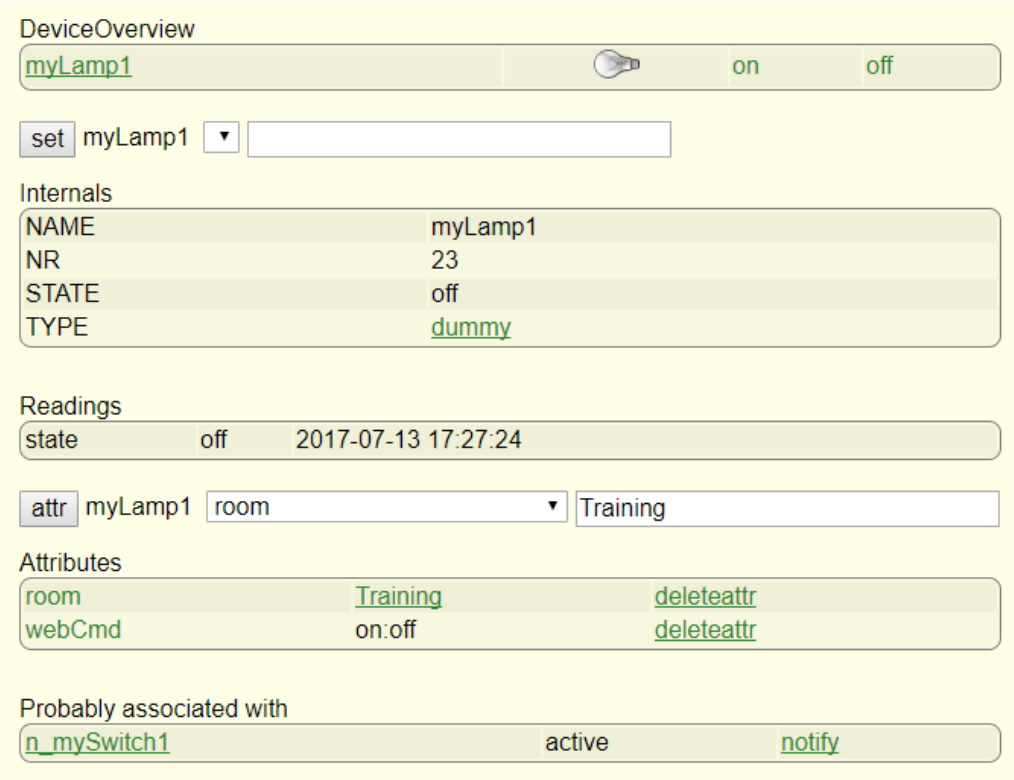

Abbildung 34: Ein beispielhaftes Device mit seinen Variablen

Devices verfügen über drei Arten von Variablen, die zusammen den Zustand ergeben, in dem sich ein Device zu einem bestimmten Zeitpunkt befindet. Diese lauten wie folgt: [107]

- **Internals** enthalten ganz grundlegende Informationen zu dem device. Sie werden vom Nutzer in den seltensten Fällen direkt bearbeitet (meist nur bei der Einrichtung des Devices).
- **Attribute** sollen das Verhalten des Devices steuern. Sie können vom Anwender typischerweise verändert (oder erst angelegt) werden.
- **Readings** besitzen im Gegensatz zu den beiden anderen Größen einen Zeitstempel. Hier werden üblicherweise Messwerte des Gerätes erfasst.

Da weder Internals noch Readings beeinflusst werden (sollen), sind Attribute diejenigen Eigenschaften eines Devices, welche man anpassen kann und somit die Funktionsweise eines Devices verändern kann. Dies erfolgt in der Eingabemaske mit dem folgenden Befehl:

```
attr <device-name> <attribut-name> <attribut-wert>
```

Dasselbe kann auch im Detailfenster eines jeden Devices mit Mausklicks gemacht werden.

Um das System zu überwachen und Rückmeldung über die Funktion zu erhalten, empfiehlt es sich den sogenannten Event Monitor zu überwachen. Dieser zeichnet die Ausgabe des Systems live auf und gibt Aufschluss über etwaige Fehler. Im Device *global* kann das *verbose* Attribut zwischen 1 und 5 eingestellt werden, um die Anzahl von Log Nachrichten zu kontrollieren, wobei 1 für *nur die wichtigsten Nachrichten* und 5 für *alle Nachrichten* steht. Die Log Nachrichten werden auch in der Datei `fhem-<jahreszahl>-<monat>.log` im Verzeichnis `/opt/fhem/log` gespeichert.

Geräte können für die Übersichtlichkeit *Räumen* zugeordnet werden. Diese werden automatisch angelegt, sobald man in einem Device das Attribut *room* auf einen Wert setzt (z.B. *Küche*). Nicht zugeordnete Devices befinden sich grundsätzlich zunächst in *Unsorted*. Außerdem kann man sich alle Devices auflisten lassen, im Raum *Everything*.

2.2.3.4 Automationen

Um in FHEM Automationen anzulegen, wird ähnlich vorgegangen, wie bei Devices, nämlich per Eingabe in die Maske:

```
define <name> notify <Suchmuster> <command>
```

Man sieht, dass auch Automationen als Devices angelegt werden, da dies die grundlegende Struktur in FHEM ist. Zunächst muss also wieder ein einzigartiger Name gewählt werden. *notify* ist der Name der Name des Hilfsmoduls, mit dem alle Automationen angelegt werden. Das Suchmuster (auch Regexp = regulärer Ausdruck) ist sehr wichtig. Es ist entweder der Name des auslösenden Devices oder die Kombination aus Device und auslösendem Ereignis (*Event*) *Devicename:Event*. Die Events kann man dem Event-Monitor entnehmen. [108] Der Command ist sehr variabel und erlaubt zweierlei Arten der Syntax:

```
set <device-name> <attribut>
```

Diese Form entspricht auch der Form, die man gewöhnlich in der Eingabemaske verwendet würde. Die Alternative ist das Scripting mit Hilfe von Perl. Hierfür wird der Perl-Code in geschweifte Klammern gepackt. Ein Beispiel:

```
define lichtAn notify bewegungsMelder {
       if(Value(“temperaturSensor”) > 17) {
```

```
                fhem “set lampe on”
        }
        else {
                fhem “set lampe off”
        }
}
```

Die grundlegende Syntax entspricht Perl, wenn auch mit kleinen Abweichungen. Nichtsdestotrotz sind wichtige Strukturen wie if/else Bedingungen, Schleifen uvm. gegeben. Im Perl-Code selbst können FHEM-eigene Funktionen ausgeführt werden, also dieselbe Syntax, die auch in der Eingabemaske verwendet wird. Mit dieser Kombination bietet FHEM ein sehr starkes Werkzeug zum Schreiben von Automationsskripten.

2.2.3.5 Z-Wave

Da Z-Wave eine so beliebte Technologie ist, müssen keine Module installiert werden o.Ä. bevor mit der eigentlichen Installation begonnen werden kann. Zunächst muss der eingesteckte Z-Wave Controller in FHEM angemeldet werden. Dafür wird der folgende Befehl ausgeführt, um den Controller als Device anzulegen:

```
define <name> ZWDongle <USB-Port>@<baud-rate>
```

Anschließend ist der Controller in FHEM eingebunden und kann verwendet werden. Um nun Z-Wave Geräte in das Netzwerk einzubinden, muss der Controller in den Inklusionsmodus gesetzt werden:

```
set <controller-name> addNode on
```

Nach einem Betätigen der Z-Wave Taste auf dem Gerät sollte eine Nachricht in der Benutzeroberfläche erscheinen, die das erfolgreiche hinzufügen bestätigt. Das hinzugefügte Gerät wurde automatisch dem Raum *Z-Wave* zugeordnet, dort befinden sich alle Z-Wave Geräte (außer der Controller). Z-Wave Geräte lassen sich in FHEM sehr leicht steuern, indem die Parameter gesetzt werden. So kann beispielsweise der Parameter *configOnTime* gesetzt werden, um die Zeit einzustellen, welche gewartet wird, bis nach einer erkannten Bewegung wieder von *open* auf *closed* gewechselt wird. Bei einigen Parametern werden auch Beschreibungen angezeigt, die dem Benutzer bei der Konfiguration helfen.

Da in dieser Anwendung der MultiSensor 5 verwendet wird, müssen noch einige Modifikationen gemacht werden. So muss das Attribut *classes* um den Wert `BASIC` erweitert werden und der Wert `WAKE_UP` entfernt werden, da der Sensor sich im USB-Betrieb befindet.

Anschließend muss noch das Attribut *configGroup1Reports* auf den Wert `224` gesetzt werden, welches entscheidet, welche Werte der Sensor liefert. Das Attribut *configGroup1Interval* kann außerdem verändert werden, um die Frequenz der Berichte einzustellen.

2.2.3.6 HomeMatic

Unter FHEM wird empfohlen für den Betrieb von HomeMatic Geräten eine RF-Schnittstelle zu verwenden, die die Kommunikation zwischen dem Raspberry und den Geräten kontrolliert bzw. weiterleitet. Eine Verwendung mit dem hier benutzten CCU2 ist herkömmlicherweise nicht vorgesehen und benötigt zusätzliche Schritte, um sie zu ermöglichen. [109]

Zunächst müssen einige Voraussetzungen getroffen werden, die die Benennung der Geräte in der CCU2 und die Firewall Einstellungen betrifft. Im Normalfall sind diese aber schon erfüllt. Der erste Arbeitsschritt ist das Installieren notwendiger Perl Module. Um Perl Module zu installieren, wird das CPAN (*Comprehensive Perl Archive Network*) Tool verwendet. Mit dem Befehl

```
cpanm Module::name
```

werden nun die Module `RPC::XML::Server`, `RPC::XML::Client`, `Thread::Queue` und `Time::HiRes` installiert, da diese notwendig sind um den RPC(*Remote Procedure Call*)-Server zu betreiben, welcher wiederum notwendig ist, um den Datenaustausch zwischen CCU2 und FHEM zu gewährleisten. Im nächsten Schritt wird das Device für die CCU2 in FHEM angelegt. Dafür wird der bekannte Befehl verwendet:

```
define <ccu2-name> HMCCU <IP der CCU2>
```

Das FHEM HMCCU Modul muss nicht noch extra installiert werden, da es Teil der FHEM Kernmodule ist. Nachdem das notwendige Device nun angelegt ist (mit dem Namen „d_ccu“), muss es noch standardmäßig konfiguriert werden. Dafür wird der Befehl `get defaults` verwendet, woraufhin alle notwendigen Einstellungen vom CCU2 erfragt und eingetragen werden. Als nächstes muss der RPC Server als externer Server konfiguriert werden:

```
attr d_ccu ccuflags extrpc
```

Und die notwendigen Interfaces eingestellt werden:

```
attr d_ccu rpcinterfaces BidCos-Wired,BidCos-RF,CUxD,VirtualDevices
```

Nun kann der Server mit `set rpcserver on` gestartet werden und sollte nach kurzer Zeit im Raum *Unsorted* mit dem Status „running/OK“ angezeigt werden. Nun ist der CCU2 soweit betriebsbereit und kann in FHEM gesteuert werden. [110]

Um als nächstes die einzelnen HomeMatic Bauteile in FHEM zu integrieren, werden sie in der Oberfläche der CCU2 eingelernt. Anschließend müssen für die verbundenen Geräte nur noch in FHEM Devices angelegt werden, dies findet mit dem Befehl `get devicelist` statt, welcher, wie schon der Name suggeriert, die Geräteliste des CCU2 ausliest. Um beispielsweise alle Geräte, die die Namensstruktur „HM-LC[…]“ besitzen und in der CCU2 Geräteliste stehen, anzulegen, wird der folgende Befehl ausgeführt:

```
get d_ccu devicelist create ^HM-KL.* t=dev f=HM_%n room=Homematic
```

Die zusätzlichen Parameter haben die folgende Bedeutung: [110]

- t={*chn*|*dev*|*all*} - Es werden nur Kanäle (*chn*) oder Geräte (*dev*) berücksichtigt
- p=<*prefix*> - Der Text wird den Namen der neuen FHEM Devices vorangestellt
- f=<*format*> - Mit *format* kann ein Template für die FHEM Devicenamen festgelegt werden. In einem Template können folgende Platzhalter verwendet werden: *%n* = CCU Geräte- oder Kanalname, *%d* = CCU Gerätename, *%a* = CCU Geräte- oder Kanaladresse

Sobald die Geräte angelegt sind, müssen sie noch mit dem Befehl `set defaults` konfiguriert werden und sind danach vollständig von FHEM aus betriebsfähig.

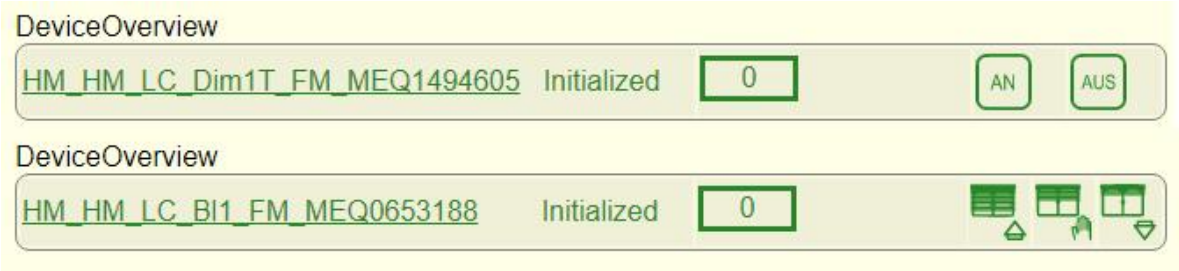

Abbildung 35: HomeMatic Steuerung in FHEM

2.2.3.7 MQTT

Um MQTT in FHEM zu integrieren wird die MQTT Bridge verwendet. Alle notwendigen Module sind schon von Anfang an verfügbar, da sie Teil der Kernmodule sind. Hierfür muss zunächst der MQTT Broker als MQTT Device angelegt.

```
define <broker-name> MQTT <IP des Brokers>:1883
```

Anschließend wird ein MQTT Bridge Device angelegt.

```
define <bridge-name> MQTT_BRIDGE <fhem-device-name>
```

Das Bridge Device wird mit einem vorhandenen Device verbunden und kann daraufhin mit den Subscribe und Publish Kanälen ausgestattet werden. Das führt dazu, dass die Readings des vorhandenen Devices an den Publish Kanal veröffentlicht und die empfangenen Nachrichten aus dem Subscribe Kanal in den Status des Devices geschrieben werden. Die Befehle dafür lauten:

```
attr <bridge-name> publishState <Publish Kanal>

attr <bridge-name> subscribeSet <Subscribe Kanal>
```

Da MQTT Nachrichten aber meistens und auch im Fall des hier verwendeten HappyBubbles Detektors in Form von JSON Objekten gesendet werden, muss der Status danach noch geparsed werden. Dafür wird das *expandJSON* Modul verwendet. Die Syntax lautet:

```
define <expandJSON-device-name> expandJSON <MQTT-device-name>.*:{.*}
(attribut1|attribut2|…)
```

Durch den regulären Ausdruck `.*:{.*}` wird der Status des Devices ausgelesen. Daraufhin werden die Readings im Device angelegt, die man ausgewählt hat (*attribut1*, *attribut2*, ...). Diese Readings können nun für Automationen ausgewertet werden. [111]

Readings

distance	0.850476094317956	2017-07-18 16:24:32
name	iBeacon	2017-07-14 19:57:01
state	{"id":"e2c56db5dffb48d2b060d0f5a71096e0_0000_0000","name":"iBeacon","distance":0.8504760943179557}	2017-07-18 16:24:32

Abbildung 36: Readings eines Devices, dem MQTT Nachrichten zugespielt werden

2.2.3.8 Umsetzung des Szenarios

Die geplante Automation besitzt einen Auslöser, eine Kondition und zwei mögliche Aktionen. In FHEM entspricht das Device vor der notify Anweisung dem Trigger und Kondition(en) und Aktion(en) werden im zweiten Teil definiert, meist in Perl.

Da die Automation durch eine Bewegung ausgelöst wird, ist der Bewegungssensor der *Trigger*. Die Automation soll also immer dann ausgeführt werden, wenn der Bewegungsmelder ausgelöst wird. Dies wird durch eine Änderung des Status auf *open* im System angezeigt. Der erste Teil der Automatisierung lautet also wie folgt:

```
define AUTOMATION notify ZWave_SENSOR_NOTIFICATION_17:open
```

„ZWave_SENSOR_NOTIFICATION_17“ ist der Name des MultiSensors in FHEM und der reguläre Ausdruck „:open“ sorgt dafür, dass die Automation nur ausgelöst wird, wenn der Status des Sensors sich auf „open“ ändert.

Als nächstes werden die Kondition und die beiden Aktionen definiert. Hierfür wird in Perl ein kurzes Skript mit einer if/else Bedingung angelegt. Die Kondition ist die Anwesenheit des Besitzers. Da FHEM keine Funktionalität liefert, die den Benutzer als *abwesend* anzeigt, wenn den Detektor keine Beacon Advertisements mehr erreichen, wird ein ähnliches Verhalten simuliert. Sobald der Beacon sich 20 Meter vom Detektor entfernt befindet, wird der Benutzer in der Automation als *abwesend* vermutet und das Verhalten ändert sich.

```
{
if (ReadingsVal("d_happybubbles ", "distance", "") > 20){
       DebianMail("benutzer\@mail.de","Alarm","Im Haus bewegt sich etwas!“)
       } else {
       fhem "set HM_HM_LC_Dim1T_FM_MEQ1494605 on";;
       fhem "set HM_HM_LC_Bl1_FM_MEQ0653188 up"
       }
}
```

Die `ReadingsVal()` Funktion wird verwendet, um Readings von Devices auszulesen. `DebianMail()` ist die Linux Mail Funktion von FHEM und nimmt die Argumente `Adressat, Betreff, Textbody(, Anhänge)` an.

2.2.4 OpenHAB

2.2.4.1 Installation

OpenHAB bietet mehrere Wege der Installation an, die sich in ihrer Komplexität und dem damit verbundenen Aufwand unterscheiden. Je nach Anwendungsfall sollte sich für einen der Wege entschieden werden. Die hier angesprochenen Installationsmöglichkeiten beziehen sich nur auf die Installation unter einem Linux System, genauer gesagt auf einem Raspberry Pi, da dieser Teil des Versuchsaufbaus ist.

Die klassische und aufwändigste Variante, welche aber auch am meisten Spielraum für Modifikationen lässt, ist die manuelle. Hier müssen sonst automatisierte Prozesse von Hand ausgeführt werden, wie das Anlegen des notwendigen Nutzerprofils mit den dazugehörigen Rechten und Eigenschaften, das Herunterladen der OpenHAB-Dateien, das Erstellen und Modifizieren des Service oder auch das Sichern oder Updaten des Betriebssystems. [112]

Eine weitaus einfachere und verbreitetere Lösung ist die paketbasierte Installation mit dem Advanced Package Tool (APT). Hierfür muss lediglich ein Repository Key zum Paketmanager hinzugefügt werden, HTTPS aktiviert werden und dann mit mithilfe von apt-get auf bekanntem Wege das openhab2 Paket installiert werden. Anschließend müssen wie auch bei der manuellen Installation Schritte befolgt werden, um den Service bereitzustellen oder das Betriebssystem zu aktualisieren, jedoch sind diese weitaus einfacher und auch weniger fehleranfällig. [112]

Sowohl die manuelle, als auch die paketbasierte Installation setzt voraus, dass auf der SD-Karte bereits das Betriebssystem Raspbian installiert ist. Weiterhin muss vor Beginn der Installation von openHAB Java installiert werden. Zwar stell Raspbian bereits eine Version von Java zu Verfügung, jedoch liegt diese unter der von openHAB empfohlenen (Java 8, Revision „101“). [113] Es muss also gegebenenfalls noch eine neuere Version von Java installiert werden.

Für den Raspberry Pi gibt es, seit der Veröffentlichung von openHAB 2, eine noch simplere Variante, um openHAB zu installieren: openHABian. Hierbei handelt es sich um eine selbstkonfigurierende Raspbian Version, die es dem Nutzer erleichtert mit openHAB zu starten. [114] Im Gegensatz zu den anderen beiden Installationsmöglichkeiten, wird nicht vorher Raspbian installiert, sondern die openHABian ISO Datei wird direkt auf die SD Karte geflasht. Anschließend muss nur noch die SD Karte in den Pi eingefügt werden und die Installation führt sich von selbst aus. Auch Prozesse wie das Aktualisieren oder Sichern des

Betriebssystems o.Ä. sind in openHABian erleichtert, da das Programm hierfür Werkzeuge bereitstellt. [115]

Für dieses Projekt wurde auf die Installation mit dem APT zurückgegriffen. Grund hierfür ist, dass diese Möglichkeit auch von anderen Systemen verwendet wird und somit eine bessere Vergleichsgrundlage geschaffen wird. Nichtsdestotrotz wird die Existenz eines Tools, welches die Installation vereinfacht, berücksichtigt und findet in der anschließenden Bewertung der Systeme Beachtung.

2.2.4.2 Oberfläche

Die openHAB Oberfläche kann nach der Installation unter der URL `<Raspberry Pi IP>:8080` aufgerufen werden. Sie sieht folgendermaßen aus:

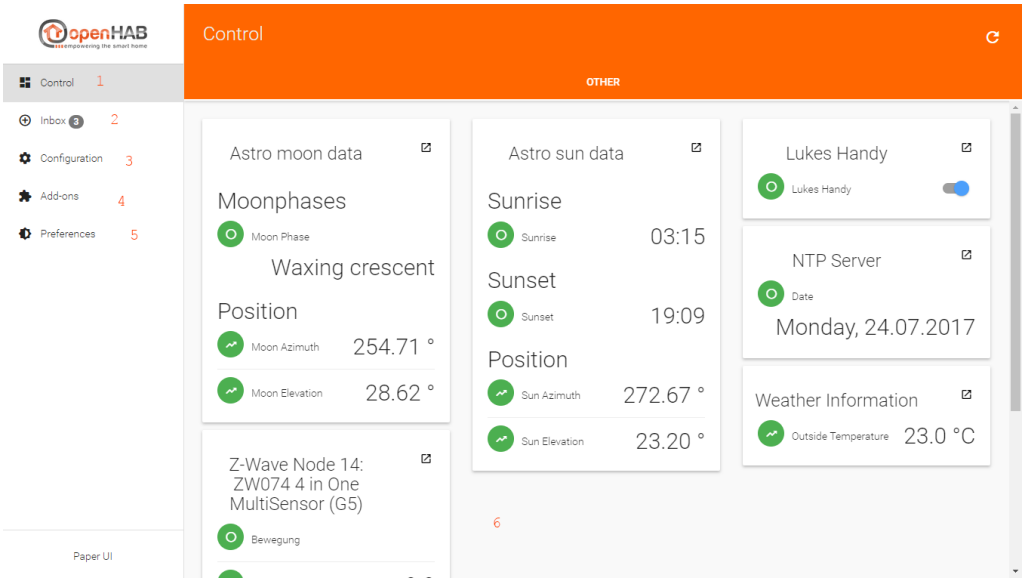

Abbildung 37: Die openHAB Benutzeroberfläche

1) Der Link zur Startoberfläche
2) Die Inbox, hier werden entdeckte Geräte angezeigt
3) Die Konfiguration, mit den Unterpunkten: *System*, *Bindings*, *Services*, *Things*
4) Im Add-Ons Menü können alle weiteren, benötigten Dinge installiert werden, wie *Bindings*, *Tranformationen* oder *Aktionen*
5) Einstellungen zum UI selbst
6) Der Hauptbereich, wo *Channel Links* angezeigt werden

openHAB hat die Besonderheit, dass nicht nur eine Benutzeroberfläche verwendet werden kann. Während die hier gezeigte Oberfläche meistens für administrative Aktionen verwendet wird, gibt es auch noch *Sitemaps*, die selbst angelegt werden können, *HABmin*, eine Abwandlung des PaperUIs mit mehr Möglichkeiten und einer anderen Optik und das *Habpanel*, eine andere Art von anlegbarer Sitemap. Je nach Anwendungsfall sollten die verschiedenen Oberflächen genutzt werden. [116]

2.2.4.3 Konfiguration und Betrieb

Um grundsätzlich Technologien in openHAB einzubinden, werden Bindings verwendet. Hierbei handelt es sich um Pakete, die den Support bereitstellen, der notwendig ist, um z.B. Z-Wave zu verwenden. Für die meisten verbreiteten Smart Home Technologien steht ein Binding zur Verfügung, dazu gehören auch die hier verwendeten Technologien Z-Wave, MQTT und HomeMatic. [117]

Bindings wurden bis zur Einführung von openHAB 2 manuell installiert, indem eine verpackte Datei heruntergeladen und deren Inhalt in den jeweiligen Ordner entpackt wurde. Anschließend mussten noch verschiedene Einstellungen angepasst werden, indem Text Dateien bearbeitet wurden. Diese Funktionalität wurde jedoch seither überarbeitet und in der Oberfläche von openHAB 2 gibt es eine Möglichkeit Bindings mit ein paar Mausklicks zu installieren. [118]

Ein essentielles Konzept, das openHAB ausmacht, ist die Trennung zwischen *Things* und *Items*. *Things* sind Einheiten, die physikalisch zum System hinzugefügt werden und potenziell mehrere Funktionalitäten auf einmal liefern können. *Items* sind die virtuelle Repräsentation der einzelnen Funktionen eines Gerätes. Somit existiert in openHAB eine strenge Trennung zwischen der physikalischen Welt (*Things*) und der Applikation, die um das Konzept der *Items* geschrieben wird (auch *virtuelle Ebene* genannt). Ein weiterer Teil dieses Konzept sind *Channel*, welche die Funktionen der *Things* repräsentieren. Der Unterschied zwischen einem *Channel* und einem *Item*, ist dass der Nutzer letzteres selbst anlegt, während die *Channel* automatisch zu jedem *Thing* mitgeliefert werden. *Channels* und *Items* werden daraufhin mit *Links* verbunden, damit eine physikalische Veränderung, wahrgenommen durch einen Sensor, aus dem *Channel* ausgelesen und an das *Item* weitergeleitet wird, damit das System damit arbeiten kann. [119, 120]

Nachdem also beispielsweise ein Sensor per Binding in openHAB integriert wurde, muss für seine Funktion ein Item erstellt werden. Items werden in der folgenden Ordnerstruktur angelegt: `/etc/openhab2/items`. Herkömmlicherweise legt man eine Datei namens `default.items` an und lagert alle Items dort, jedoch ist man in dieser Hinsicht nicht

eingeschränkt. Solang die Dateiendung `.items` verwendet wird, erkennt openHAB die Items. Die Struktur für das Anlegen von Items lautet wie folgt: [121]

```
ItemType ItemName "ItemDescription" <ItemIcon> { ItemToThingChannelLink }
```

Es gibt verschiedene *Item Types*, die jeweils verschiedene Eigenschaften besitzen. Ein Item des Typs *Dimmer* speichert seinen Zustand zum Beispiel als Prozentzahl, während ein Item des Typs *Switch* seinen Zustand binär speichert (*on/off*). Darüber hinaus gibt es viele weitere Item Types, die für verschiedene Anwendungsfälle verwendet werden können, wie *Color*, *Rollershutter* oder *Number*. [120]

Der nächste Schritt ist das Anlegen einer *Sitemap*. Hierbei handelt es sich um eine separate Ansicht der Frontendoberfläche, welche frei konfigurierbar ist. Diese eignen sich sehr gut zur Steuerung des Smart Home Systems. Sitemaps werden in der Ordnerstruktur `/etc/openhab2/sitemaps` angelegt und verhalten sich gleich wie Items, in Hinsicht auf die Benennung der Dateien. Die Syntax einer `.sitemap` Datei lautet: [121]

```
sitemap default label="Sitemap Titel"
{
    ElementType item=ItemName1 label="Beschreibung des Items 1"
    ElementType item=ItemName2 label="Beschreibung des Items 2"
}
```

In Sitemaps können Items als *Elemente* angelegt werden und haben dementsprechend verschiedene *Element Typen*. Diese sind nicht zu verwechseln mit *Item Typen*. Element Typen beschreiben einzig die Art und Weise, wie Elemente in der Sitemap angezeigt werden und was geschieht, wenn mit ihnen interagiert wird. Ein *Slider* Element wird als ein Schiebepunkt Regler angezeigt und liefert eine Prozentzahl zurück, weshalb er sich gut für Items des Typen *Dimmer* eignen. Ein *Switch* Element hingegen wird als Schalter angezeigt und liefert binäre Werte zurück, weshalb er sich gut für Items des Typen *Switch* eignet. [122]

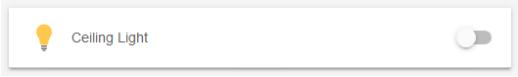

Abbildung 38: Ein Switch Element

Abbildung 39: Ein Slider Element

Weiterhin gibt es die Möglichkeit mehrere *Frames* anzulegen, um die Sitemap besser visuell zu unterteilen.

Um die Verwendung von openHAB zu vereinfachen, gibt es die Möglichkeit sich den Log *live* anzeigen zu lassen. Dafür muss die *Karaf Console* geöffnet werden. Dafür wird eine SSH Verbindung aufgebaut:

```
ssh -p 8101 openhab@localhost
```

Das Passwort für die Verbindung lautet im Werkszustand `habopen`. Die erste Verwendung kann sehr lang dauern, da zunächst die notwendigen Dateien installiert werden müssen. Im Anschluss wird dann der Befehl `log:tail` eingegeben, woraufhin der Live Log angezeigt wird. Hier findet man nützliche Fehlermeldungen, die das Debuggen erleichtern. [123]

2.2.4.4 Automationen

openHAB bietet für das Schreiben von Automationen eine spezielle Version von *Eclipse*, der integrierten Entwicklungsumgebung für die Programmiersprache Java. Der sogenannte *Eclipse SmartHome Designer* ermöglicht es, direkt auf das Dateisystem des Raspberry Pis zuzugreifen (vorausgesetzt, die Netzwerkfreigabe wurde aktiviert), Konfigurationsdateien direkt zu bearbeiten und hat den Vorteil von Syntax Highlighting, Syntaxüberprüfung und Autovervollständigung.

In openHAB werden Automationen *rules* (*Regeln*) genannt und sie haben eine zweigespaltene Struktur: Jede Regel hat eine oder mehrere *Triggerconditions* und einen *Script Block*, der ausgeführt wird, wenn die Kondition(en) erfüllt werden. Es gibt drei verschiedene Arten von Triggern: [124]

- Item(-Event)-basierte Trigger: Sie reagieren auf Ereignisse auf dem openHAB „event bus“, also zum Beispiel Anweisungen und Status Updates für Items.
- Zeit-basierte Trigger: Sie reagieren zu bestimmten Zeiten, zum Beispiel um Mitternacht, jede Stunde, etc.
- System-basierte Trigger: Sie reagieren auf gewisse System Status.

Für vernetzte Automationen interessant sind hauptsächlich die Item-basierten Trigger. Diese haben eine der folgenden Strukturen:

```
Item <item> received command [<command>]
```

`Item <item> received update [<state>]`

`Item <item> changed [from <state>] [to <state>]`

Der Script Block wird im Java Dialekt *Xtend* geschrieben. [125] Die Syntax ist sehr ähnlich zu Java, jedoch mit dem Vorteil, dass die Skripte nicht vor der Ausführung kompiliert werden müssen, da sie „at runtime“ interpretiert werden können, also sozusagen „im Laufen (des Betriebssystems)“. [124] Da es sich hier also um eine ausgeprägte Programmiersprache handelt, sind dem Entwickler kaum Grenzen gesetzt. Außerdem können am Anfang der `.rules` Datei Bibliotheken importiert werden und Variablen angelegt werden.

Die drei Teile der Regel sehen dann zusammengesetzt folgendermaßen aus:

```
import x
String str = “Hallo Welt”
rule “Regeltest”
when
	Item dummy received update
then
	if (str==”Hallo Welt”) {
		light.on();
	} else {
		light.off();
	}
end
```

Zu beachten ist die `when/then/end` Struktur der Regel. Außerdem können mehrere Regeln in die selbe Datei geschrieben werden, solang jede eine Namendeklaration mit der Struktur `rule „x“` besitzt. [124]

2.2.4.5 Z-Wave

Um Z-Wave in openHAB zu installieren, muss zu nächst das Z-Wave Binding im Add-Ons Menü installiert werden. Daraufhin wird in der Inbox das „+“ gedrückt und anschließend das Z-Wave Binding ausgewählt. Anschließend muss der USB-Pfad des Controllers angegeben werden, im Normalfall `/dev/ttyUSB0`. Anschließend fängt der Controller sofort an, sich nach Geräten umzuschauen. Es sollte dann auch sofort der MultiSensor in der Inbox auftauchen. Wenn dies nicht geschieht, muss er nachträglich über die Inbox hinzugefügt werden (ähnlich wie der Controller). Im HABmin Menü kann sehr einfach auf die Parameter der Z-Wave Komponenten zugegriffen werden und so das Z-Wave Netzwerk genau so konfiguriert

werden, wie es notwendig ist. Um die Z-Wave Sensordaten in der Startoberfläche von PaperUI angezeigt zu bekommen, müssen die zuständigen Channels noch im MultiSensor Thing aktiviert werden.

2.2.4.6 HomeMatic

Der einzige notwendige Schritt für die Einbindung von HomeMatic in openHAB ist das Aktivieren des HomeMatic Bindings und daraufhin das Hinzufügen der HomeMatic Geräte – für die CCU2 muss die IP-Adresse des Geräts angegeben werden, die anderen Komponenten werden daraufhin automatisch erkannt. Voraussetzung ist natürlich, dass die Geräte in der CCU2 registriert sind. Damit ist die Einbindung von HomeMatic in openHAB sehr simpel und birgt kaum Fehlerquellen.

2.2.4.7 MQTT

Um MQTT in openHAB umzusetzen müssen zunächst in die Datei `/etc/openhab2/services/mqtt.cfg` die Daten des Brokers geschrieben werden. Wirklich relevant sind nur die folgenden Zeilen, vorausgesetzt man verwendet keine passwortgeschützte Verbindung:

```
broker.url=tcp://localhost:1883

broker.clientId=openHAB
```

Wichtig ist, die Benennung des Brokers, da diese später noch verwendet werden muss. In diesem Fall heißt der Broker einfach `broker`. In der ersten Zeile wird die Adresse des Brokers angegeben – diese ist in diesem Fall `localhost`, da der Broker auf demselben Gerät läuft wie openHAB selbst. Der Port `1883` ist standardmäßig für MQTT vorgesehen.

Als nächstes müssen die Nachrichten, die der Broker empfängt und verteilt auf irgendeine Weise entgegengenommen werden. Hierfür wird ein Item angelegt, das mit dem für HappyBubbles relevanten Kanal verbunden ist (`happy-bubbles/presence/ha/happy-bubble`). Die Syntax hierfür lautet:

```
String distance "Distanz[%s]" <line> {mqtt="<[broker:happy-
bubbles/presence/ha/happy-bubble:state:JSONPATH($.distance)]"}
```

Einige Dinge sind zu beachten:

- `String` ist der Typ des Items. Zwar handelt es sich um eine Distanz, also eine Zahl, für unsere Zwecke genügt es aber, den Wert als String zu speichern
- `distance` ist der Name des Items
- `"Distanz[%s]"` ist das Label des Items, also was angezeigt wird. [%s] ist die Variableninjektion in openHAB, hier wird also die Distanz selbst angezeigt
- `<line>` ist das Symbol, das in der Sitemap angezeigt wird

Der Code zwischen den geschweiften Klammern ist so zu deuten:

- `mqtt` teilt openHAB mit, dass es sich um MQTT Nachrichten vom Broker handelt
- `<` bzw. `>` teilen dem System mit, ob Nachrichten hereinkommen oder raus geschickt werden sollen. In diesem Fall soll auf die Nachrichten des Präsenzdetektors gehört werden
- `broker` ist der vorhin schon angesprochene Name des Brokers
- Danach folgt der übliche Kanal, in den die Nachrichten des Detektors veröffentlicht werden
- `state` teilt dem System mit, dass die Nachricht in den `state` des Items geschrieben werden soll
- Da es sich bei den Veröffentlichungen des Detektors um JSON Objekte handelt, müssen diese erst geparsed werden, damit im `state` des Items nur die eigentliche Distanz landet. Hierfür muss vorab noch die `JSONPATH` Transformation im Add-On Menü von PaperUI installiert werden

Anschließend muss noch eine Sitemap angelegt werden, bzw. das neue Item in eine existierende Sitemap eingetragen werden:

```
Text item=distance
```

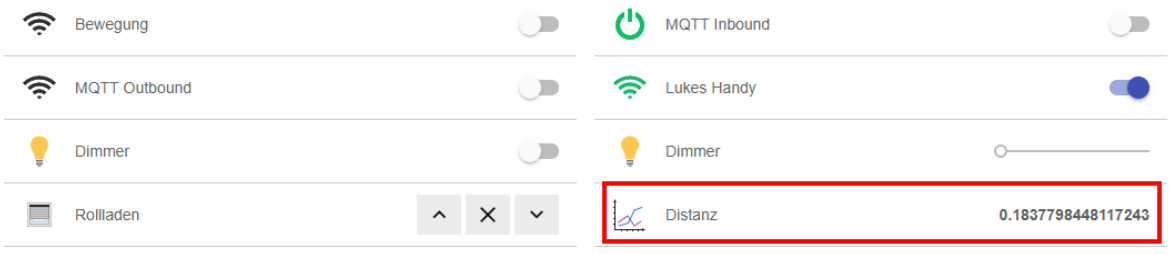

Abbildung 40: Die MQTT Distanz Anzeige in einer Sitemap

2.2.4.8 Umsetzung des Szenarios

Um die Automation zu schreiben muss zunächst eine Regel im Ordner `/etc/openhab2/rules/` angelegt werden. Der Auslöser der Automation ist eine vom Bewegungsmelder registrierte Bewegung. In openHAB wird diese Item-basierte Event folgendermaßen abgefangen:

```
Item motion_sensor changed from OFF to ON
```

Im Script Block soll das folgende Verhalten erreicht werden: Wenn die vom Detektor übermittelte Distanz über 20 liegt, soll eine Mail gesendet werden. Wenn sie unter 20 liegt, soll der Dimmer angeschaltet werden. Die Distanz 20 ist zwar arbiträr, jedoch verfügt openHAB nicht über die Funktionalität festzustellen, wann der Detektor den Beacon als *away* meldet. Da es sich um einen Java Dialekt handelt, der im Script Block verwendet wird, kann eine einfache if/else Bedingung geschrieben werden. Zunächst muss jedoch die als String gespeicherte Distanz zu einer Variable des Typs *Double* geparsed werden, damit auf die Variable Vergleichsoperatoren angewendet werden können. Dafür muss zusätzlich das Java Paket `java.lang.Double` importiert werden. Um die Mail Funktionalität zu ermöglichen muss außerdem die *Mail* Action installiert werden (Add-On Menü) und die folgenden Zeilen in die Datei `mail.cfg` im Ordner `/etc/openhab2/services/` geschrieben werden:

```
hostname=<smtp-server des mail providers>
port=587
username=<username>
password=<password>
from=<absender adresse>
tls=true
```

Der fertige Script Block sieht dann folgendermaßen aus:

```
var String distanceVar = distance.state
var Double distanceDouble = Double.parseDouble(distanceVar)
if (distanceDouble < 20)
	sendCommand(dimmer, 100)
else
	sendMail("lukasm2411@gmail.com", "Intruder Alert", "Es ist niemand daheim, aber es hat sich was bewegt!")
```

2.3 Vergleich

2.3.1 Installationsaufwand

Plattform:

Bei allen drei Plattformen ist selbst die „manuelle“ Installation sehr simpel. Home-Assistant bietet mit dem All-In-One Installationsskript eine Möglichkeit, die es dem Nutzer erlaubt mit nur einem Kommandozeilenbefehl die gesamte Installation auszuführen, was sehr komfortabel ist. Das äquivalente Prozedere beinhaltet bei openHAB schon weitaus mehr Befehle, die von Hand ausgeführt werden müssen, wodurch die Fehlerquote steigen kann. Jedoch ist die Installation auch in diesem Fall noch sehr simpel und mit ein wenig Linux Erfahrung sollte sie kein Problem darstellen. Bei FHEM ist die Installation per APT auch möglich und liegt damit mit der Komplexität ungefähr in der Mitte. Nichtsdestotrotz sind alle drei Installationsvorgänge sehr einfach und erlauben auch weniger erfahrenen Nutzer einen leichten Einstieg. Home-Assistant und openHAB bieten mit ihren eigenen Raspbian Varianten, Hassbian und openHABian, einen noch einfacheren Weg, die Serversoftware zu installieren und schneiden darum in dieser Kategorie noch ein wenig besser ab. Leider verfügt FHEM nicht über eine vergleichbare Software.

Home-Assistant	openHAB	FHEM
++	++	+

Z-Wave:

Da Z-Wave eine der führenden Technologien auf dem Smart Home Markt ist, gestaltet sich die Anbindung an alle drei Systeme ähnlich einfach. Zwar kann es einige Zeit dauern, bis der Nutzer die Funktionsweise von Z-Wave *selbst* verstanden hat, das ist aber nicht die Schuld der Plattformen. Diese liefern schon im Werkszustand alle die notwendigen Werkzeuge, um Z-Wave schnell zu installieren. Die Anbindung an FHEM ist marginal umständlicher, da noch bestimmte Parameter geändert werden müssen, daher schneidet diese Plattform hier am schlechtesten ab. Home-Assistant und openHAB bieten mit ihren jeweiligen Programmen *OpenZwave Control Panel* und *HABmin* starke Werkzeuge, um Z-Wave gut zu kontrollieren,

jedoch ist dasselbe Level von Kontrolle auch in FHEM möglich, ohne zusätzliche Plattformen zu nutzen.

Home-Assistant	openHAB	FHEM
++	++	+

HomeMatic:

Sowohl in Home-Assistant als auch in openHAB ist die HomeMatic Installation sehr einfach. Sobald man die IP-Adresse der CCU2 angibt, wird dessen Register ausgelesen und die eingetragenen Geräte sofort eingebunden. Der Installationsaufwand ist somit fast gleich null. Im Gegensatz dazu ist die Installation in FHEM leider sehr aufwändig und setzt gewisse Programmiererfahrungen voraus, da unter anderem auch Perl Module installiert werden müssen. Somit hat FHEM hier einen großen Nachteil, obgleich nach der umständlichen Installation alles sehr gut funktioniert hat.

Home-Assistant	openHAB	FHEM
++	++	-

MQTT:

Die Integrierung der HappyBubbles MQTT Technologie gestaltet sich in Home-Assistant relativ einfach und bietet darüber mit der Away-Time ein Feature, das den Umgang mit der Technologie sinnvoller und stimmiger macht. Dagegen ist sowohl die Installation und auch der Betrieb von MQTT in openHAB und FHEM unterlegen. In beiden Fällen muss der Benutzer eine umständlichere Installation durchführen und auch JSON Parsing betreiben, bevor er die gewünschten Werte korrekt übermittelt bekommt. Hier hat Home-Assistant einen großen Vorteil.

Home-Assistant	openHAB	FHEM
++	-	-

2.3.2 Support

Alle drei Plattformen haben als Open Source Projekte eine sehr große Sammlung an Ressourcen, die es dem Nutzer erleichtern, Probleme zu beheben, Technologien einzubinden oder Projekte umzusetzen. Home-Assistant verfügt trotzdem über die gefühlt größte und zuverlässigste Hilfscommunity. Nicht nur stehen extrem viele Information auf der Webseite bereit, die von der Community gepflegt wird, mit zusätzlichen Ressourcen wie der *Reddit* Community oder dem Chatkanal auf der Plattform *Discord* kommt man hier in jedem Fall sehr schnell zu einer Lösung (vgl. 1.2.6.1.1). Auch openHAB verfügt über ein aktives Hilfsforum und die Dokumentation ist umfangreich, wenn auch an Stellen veraltet. An diesem Problem leiden aber grundsätzlich alle Open Source Projekte, auch Home-Assistant und FHEM. FHEM verfügt über ein eigenes Wiki, welches sehr umfangreich wirkt und über sehr detaillierte Anweisungen verfügt, was es gegenüber den anderen beiden Plattformen hervorhebt, welche zwar auch Anleitungen bereitstellen, diese sind aber eher spärlich und meistens sehr allgemeingültig geschrieben.

Home-Assistant	openHAB	FHEM
++	o	o

Zwar ist das Level von Support, das openHAB und FHEM bereitstellen durchaus gut, jedoch sind sie im Vergleich zu Home-Assistant nicht ganz so gut.

2.3.3 User Experience

Sowohl Home-Assistant als auch openHAB haben eine moderne Oberfläche, da beide das Material Design von Google verwenden, welches als einer der modernen Frontend Standards gilt. FHEM hingegen wirkt sehr alt und rudimentär. Zwar hat das für einen Entwickler per se keinen großen Einfluss darauf, wie effizient er mit dem Programm umgehen kann, jedoch wäre ein Upgrade von Vorteil.

openHAB hat mit seinen unterschiedlichen Oberflächen einen großen Vorteil, da es dem Nutzer ermöglicht das Interface frei zu wechseln, je nachdem was gerade benötigt wird. Außerdem ist das Zusammenstellen von Sitemaps ein sehr nützliches Feature, da dem Nutzer überlassen wird, wie er seine Smart Home Steuerung aussehen lassen möchte bzw. wie er seine Smart Home Geräte steuert.

Ein wichtiger Aspekt ist die Schnelligkeit der Plattformen. Zwar sind alle drei im normalen Umgang gleich schnell, da die auszuführenden Aktionen meist sehr simpel sind und der vorliegende Raspberry Pi 3 relativ leistungsstark ist, jedoch gibt es doch Unterschiede. Um beispielsweise manche Funktionen von openHAB zu verwenden, müssen Bindings oder Actions installiert werden. Dies kann teilweise sehr lang dauern und führt manchmal dazu, dass die Oberfläche sich aufhängt. Nach einem Reload der Seite sind die Pakete dann meist installiert, nichtsdestotrotz führt das zu einem gewissen Disconnect in der Benutzerführung, was einen negativen Einfluss hat. Außerdem müssen alle drei Plattformen bei wichtigen Änderungen neugestartet werden, sprich der Linux Service muss manuell neugestartet werden (`systemctl`). Das ist an sich kein Problem, jedoch dauert dieses Neustarten je nach Plattform kürzer oder länger. In Home-Assistant dauert der Vorgang ungefähr eine Minute, während er in openHAB teilweise doppelt so lang braucht. FHEM führt den Neustart dank seiner schlanken Bauart sehr schnell aus.

Home-Assistant	openHAB	FHEM
+	++	-

3 Schluss

3.1 Zusammenfassung

Im Folgenden soll dargestellt werden, wie ich beim Erstellen dieser Arbeit vorgegangen bin.

Die Idee Open Source Plattformen für die Smart Home Entwicklung zu vergleichen war von Anfang an fest, jedoch standen einige andere Dinge nicht fest, die erst im Laufe der Arbeit entschieden wurden. Dazu gehört das exemplarische Szenario, das zum Vergleich der Plattformen genutzt werden sollte. Welche Sensorentypen sollen verwendet werden? Welche Aktorentypen? Und danach die Frage, welche tatsächlichen Sensoren-/Aktorenprodukte gekauft werden sollten. Im Endeffekt habe ich mich dazu entschieden die Komplexität des Szenarios relativ gering zu halten, da allein die Einarbeitung in die verschiedenen Technologien und Plattformen einen sehr großen Aufwand darstellte. Die Frage nach den Sensoren sollte auch erst geklärt werden, nachdem ich die Recherche schon hinter mir hatte, da ich nicht irgendwelche Geräte kaufen wollte. Nachdem ich mich in das Thema der verschiedenen Protokolle eingearbeitet hatte, konnte ich dann eine durchdachtere Entscheidung treffen.

Eine weitere Entscheidung, die erst im Laufe der Recherche getroffen wurde, war Home-Assistant überhaupt in meine Arbeit mitaufzunehmen. Ursprünglich war das Hauptaugenmerk auf openHAB, FHEM und Node-RED. Jedoch wurde mir, während ich auf der Suche nach geeigneten Sensoren und Aktoren war, von vielen Seiten die Plattform Home-Assistant ans Herz gelegt, woraufhin ich beschloss, sie zu untersuchen. Node-RED fiel dann schließlich aus dem Vergleich komplett heraus, da es verglichen mit den anderen Plattformen sehr wenig eigentliche praktische Nutzung sieht und sich daher ein Vergleich als nicht sehr sinnvoll herausgestellt hat.

Die erste Umsetzung des Szenarios und damit auch der erste Kontakt mit Sensoren, Aktoren, einer Plattform etc. fanden in Home-Assistant statt. Daher fiel mir ein ausgewogener Vergleich der Plattformen in manchen Hinsichten auch schwer, da ich weitaus mehr Erfahrung mit Home-Assistant hatte als mit den anderen zwei. Grund dafür ist, dass die größten Schwierigkeiten nicht die Plattformen selbst, sondern die Geräte inklusive ihrer Protokolle, also besonders Z-Wave und MQTT, gemacht haben. Sobald deren Funktionsweise einmal verstanden ist, ist das Umgehen mit den Plattformen selbst eigentlich simpel und dazu vom Aufwand her untereinander sehr ähnlich. Ein erfahrener Smart Home Tüftler hätte wahrscheinlich für alle drei Plattformen gleich lang gebraucht, da ich jedoch zunächst die zu meinem Szenario gehörenden Technologien ergründen musste und dies mit

der ersten Verwendung von Home-Assistant einherging, dauerte dessen Einarbeitung am längsten.

Die drei Plattformen wurden auf drei verschiedenen SD-Karten installiert, wodurch ein fliegender Wechsel zu jeder Zeit möglich war, was sich auch als sehr nützlich herausgestellt hat. Sobald die Technologien überall gut eingebunden sind, muss praktisch nichts mehr getan werden, außer die Karten zu wechseln, um beispielsweise von openHAB auf FHEM umzuspringen. Der Raspberry Pi hat sich im Laufe der Projektumsetzung als ein sehr zuverlässiges Gerät erwiesen, welches auch gerne mehrere Wochen am Stück ohne Probleme betrieben werden kann und sich auch durch Fall- oder Wasserschäden wirklich beeinträchtigen lässt.

3.2 Fazit und Ausblick

Abschließend lässt sich sagen, dass es im Laufe der Arbeit keine Überraschungen gab, was die Eignung der drei Plattformen für die Umsetzung eines Smart Home Projekts angeht. Das sollte keine Überraschung sein, da sie dafür konzipiert sind. Sehr große Unterschiede darin, wie gut sich die einzelnen Plattformen in verschiedenen Disziplinen verhalten, sind nicht zu finden, diese sind eher marginal. Zwar gibt es durchaus Stellen, an denen sich die eine Plattform für eine bestimmte Technologie besser eignet, jedoch gibt es für alles eine Lösung. Daher hat auch jeder Teil der Umsetzung funktioniert, was nicht notwendigerweise selbstverständlich ist.

Nichtsdestotrotz stellt sich die Frage, welche Plattform am „besten“ ist. Da die Antwort offensichtlich nicht in der Umsetzung an sich liegt, muss an anderen Stellen gesucht werden. Der meiner Meinung größte Faktor, den eine Plattform über die anderen stellt, ist die Verbreitung und das damit verbundene Wachstum der Plattform. Also auch: Wie wahrscheinlich ist es, dass eine Plattform sich zum Marktführer durchsetzt und dadurch auch in Zukunft noch weiterentwickelt wird? Und hier sticht ganz klar Home-Assistant heraus. Zwar existieren FHEM und openHAB schon länger, jedoch erkennt man an den Statistiken der jeweiligen Hilfeforen, dass Home-Assistant bessere Zahlen als openHAB in den Kategorien „letzte 7 Tage“ und „letzte 30 Tage“ aufweisen kann. FHEM verfügt leider nicht über diese Statistik, jedoch hat FHEM allein deshalb nicht das Potenzial eine so große Rolle einzunehmen, weil es sehr deutschsprachig ist. Es existieren zwar auch Dokumentationen auf Englisch und einige englischsprachige Nutzer, trotzdem kann es kaum mit internationalen Projekten wie Home-Assistant oder openHAB gleichgestellt werden.

Während openHAB sich mit der Entwicklung des Eclipse SmartHome Frameworks in eine Richtung entwickelt, wo professionelle Projekte damit entstehen können, hat wird Home-Assistant sich wahrscheinlich in Zukunft zum Spitzenreiter im Bereich der privaten Hausautomatisierung durchsetzen. Dafür spricht auch die Entwicklung der neuen Plattform Hass.io, die die Entwicklung mit Home-Assistant noch einmal vereinfacht und die Plattform somit noch attraktiver für Entwickler aller Art macht.

Allgemein lässt sich über das Thema Smart Home sagen, dass momentan ein spannender Zeitpunkt in der Entwicklung des Marktes ist. Es scheint so, als würden sich allmählich Favoriten auf einem Markt herauszukristallisieren, der bisher noch sehr ungefestigt war. Sowohl an der Protokollfront, als auch bei den Plattformen selbst, scheinen die Konsumenten inzwischen Produkte gefunden zu haben, die es Wert sind weiterentwickelt zu werden, im Fall von Open Source Projekten auch von den Konsumenten selbst. Trotzdem bleibt es spannend zu sehen, was mit komplett von Communities getragenen Projekten passieren. Interessanterweise laufen alle drei Plattformen noch unter der Leitung ihrer Erfinder. Besonders wenn sich das ändert, könnte es spannende Entwicklungen geben.

Nichtsdestotrotz fühlt es sich momentan noch so an, als wäre die Hausautomatisierung, besonders unter Open Source Plattformen, eine den Entwicklern vorbehaltene Kunst, die noch einiges an Reifezeit benötigt, um von herkömmlichen Benutzern wirklich wertgeschätzt werden zu können. Es benötigt schlichtweg noch zu viel Hintergrundwissen, um die Technologien zum Laufen zu bekommen, als dass eine Person ohne diese Erfahrungen dazu in der Lage wäre. Das heißt nicht, dass Smart Home nicht schon im Mainstream angekommen ist. Proprietäre Plattformen wie SmartThings ermöglichen es jetzt schon technisch nicht-versierten Leuten, ihr Haus zu automatisieren. Jedoch hat das eben seinen Preis und seine Einschränkungen. Doch bis Home-Assistant und co. es nicht geschafft haben, den Großteil der „Coding“-Arbeit aus der Bedienung ihrer Systeme zu entfernen, muss sich der Großteil der Konsumenten wohl noch damit zufriedengeben.

4 References

[1] C. DiBona, S. Ockman, and M. Stone, *Opensources: Voices from the open source revolution / edited by Chris DiBona, Sam Ockman & Mark Stone*. Beijing, London: O'Reilly, 1999.

[2] Oxford Dictionaries, *smart home - definition of smart home in English | Oxford Dictionaries.* [Online] Available: https://en.oxforddictionaries.com/definition/smart_home. Accessed on: Jul. 11 2017.

[3] Wikipedia, *Smart Home.* [Online] Available: https://de.wikipedia.org/w/index.php?oldid=167145484. Accessed on: Jul. 11 2017.

[4] Manuel Raffel, *openHAB – Empowering the Smart Home - History, Concepts, Examples.* [Online] Available: http://wi.wu-wien.ac.at:8002/rgf/diplomarbeiten/Seminararbeiten/2014/201406-Raffel-OpenHAB.pdf. Accessed on: Jul. 11 2017.

[5] R. Harper, *Inside the smart home*. London, New York: Springer, 2003.

[6] N. K. Suryadevara and S. C. Mukhopadhyay, *Smart Homes: Design, Implementation and Issues*. Cham: Springer International Publishing, 2015.

[7] L. Brand, T. Hülser, V. Grimm, and A. Zweck, *Internet der Dinge: Perspektiven für die Logistik: Übersichtsstudie.* [Online] Available: https://www.vdi.de/fileadmin/vdi_de/redakteur/dps_bilder/TZ/2009/Band%2080_IdD_komplett.pdf. Accessed on: Jul. 17 2017.

[8] Wikipedia, *Bewegungsmelder.* [Online] Available: https://de.wikipedia.org/w/index.php?oldid=164896750. Accessed on: Jul. 17 2017.

[9] Wikipedia, *Temperatursensor.* [Online] Available: https://de.wikipedia.org/w/index.php?oldid=165745781. Accessed on: Jul. 17 2017.

[10] Wikipedia, *Hygrometer.* [Online] Available: https://de.wikipedia.org/w/index.php?oldid=160221374. Accessed on: Jul. 17 2017.

[11] *6 Smart Interior Air Quality Monitors You Should Buy For Your Home.* [Online] Available: http://www.makeuseof.com/tag/6-smart-interior-air-quality-monitors-you-should-buy-for-your-home/. Accessed on: Jul. 17 2017.

[12] *VOC Sensors and Monitors - Volatile Organic Compounds.* [Online] Available: http://www.aeroqual.com/sensors/voc-sensors-monitors. Accessed on: Jul. 17 2017.

[13] Wikipedia, *Luftgütesensor.* [Online] Available: https://de.wikipedia.org/w/index.php?oldid=153925477. Accessed on: Jul. 17 2017.

[14] *Core concepts | Philips Hue API.* [Online] Available: https://www.developers.meethue.com/documentation/core-concepts. Accessed on: Jul. 17 2017.

[15] R. Dillet, *Philips Hue releases new connected light bulbs for different shades of white light.* [Online] Available: https://techcrunch.com/2016/03/14/philips-hue-releases-new-connected-light-bulbs-for-different-shades-of-white-light/. Accessed on: Jul. 26 2017.

[16] Wikipedia, *Chromecast - Wikipedia.* [Online] Available: https://en.wikipedia.org/w/index.php?oldid=787631038. Accessed on: Jul. 17 2017.

[17] Wikipedia, *Apple TV - Wikipedia.* [Online] Available: https://en.wikipedia.org/w/index.php?oldid=791421814. Accessed on: Jul. 26 2017.

[18] M. Richardson and S. P. Wallace, *Getting started with Raspberry Pi*. San Francisco, CA: Maker Media, 2016.

[19] *Raspberry Pi Zero out now. Get yours free with The MagPi #40! - The MagPi Magazine.* [Online] Available: https://www.raspberrypi.org/magpi/raspberry-pi-zero-out-today-get-it-free-with-the-magpi-40/#. Accessed on: Apr. 08 2017.

[20] *File:RaspberryPi 3B.svg - Wikimedia Commons.* [Online] Available: https://commons.wikimedia.org/wiki/File%3ARaspberryPi_3B.svg. Accessed on: May 13 2017.

[21] *Raspberry Pi 3 Model B - Raspberry Pi.* [Online] Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed on: Apr. 16 2017.

[22] *Raspberry Pi or Arduino? One Simple Rule to Choose the Right Board | Make:*. [Online] Available: http://makezine.com/2015/12/04/admittedly-simplistic-guide-raspberry-pi-vs-arduino/. Accessed on: Apr. 16 2017.

[23] J. M. Hughes, *Arduino: A technical reference : a handbook for technicians, engineers, and makers*. Sebastopol, CA: O'Reilly Media, Inc, 2016.

[24] *File:Arduino Uno - R3.jpg - Wikimedia Commons.* [Online] Available: https://commons.wikimedia.org/wiki/File:Arduino_Uno_-_R3.jpg. Accessed on: May 31 2017.

[25] *Arduino.* [Online] Available: https://store.arduino.cc/. Accessed on: Apr. 16 2017.

[26] Wikipedia, *List of Arduino boards and compatible systems - Wikipedia.* [Online] Available: https://en.wikipedia.org/w/index.php?oldid=776400104. Accessed on: May 13 2017.

[27] M. Banzi, M. Shiloh, and B. Jepson, *Getting started with Arduino*. Sebastopol, California: Maker Media, 2015.

[28] *Arduino - FAQ*. [Online] Available: https://www.arduino.cc/en/Main/FAQ#toc13. Accessed on: May 29 2017.
[29] Wikipedia, *Netzwerkprotokoll*. [Online] Available: https://de.wikipedia.org/w/index.php?oldid=164927737. Accessed on: Jul. 12 2017.
[30] Wikipedia, *OSI-Modell*. [Online] Available: https://de.wikipedia.org/w/index.php?oldid=166935826. Accessed on: Jul. 12 2017.
[31] S. Goodwin, *Smart Home Automation with Linux and Raspberry Pi*. Berkeley, CA, s.l.: Apress, 2013.
[32] *Z-Wave the Smartest Choice for your Smart Home*. [Online] Available: http://www.z-wave.com/faq. Accessed on: Jun. 28 2017.
[33] *Member Companies of the Z-Wave Alliance - Z-Wave Alliance*. [Online] Available: http://z-wavealliance.org/z-wave_alliance_member_companies/. Accessed on: Jun. 28 2017.
[34] O. Hersent, D. Boswarthick, and O. Elloumi, *The internet of things: Key applications and protocols / Olivier Hersent, David Boswarthick, Omar Elloumi*. Chichester, West Sussex: Wiley, 2012.
[35] *OZW Utilities*. [Online] Available: http://www.openzwave.com/home/. Accessed on: Jun. 28 2017.
[36] Wikipedia, *ZigBee*. [Online] Available: https://de.wikipedia.org/w/index.php?oldid=164933539. Accessed on: Jul. 12 2017.
[37] G. Shi and K. Li, *Signal Interference in WiFi and ZigBee Networks*. Cham: Springer International Publishing; Imprint: Springer, 2017.
[38] R. K. Markus Kraue, *Drahtlose zigbee-netzwerke*: Springer Vieweg, 2014.
[39] *BidCoS: Funkstandard bei HomeMatic - Funkprotokoll für die Hausautomation*. [Online] Available: https://www.homeandsmart.de/bidcos-funkstandard-eq-3-hausautomation. Accessed on: Jul. 27 2017.
[40] Wikipedia, *Bluetooth Low Energy - Wikipedia*. [Online] Available: https://en.wikipedia.org/w/index.php?oldid=781355481. Accessed on: Jul. 12 2017.
[41] K. Townsend, C. Cufí, Akiba, and R. Davidson, *Getting started with Bluetooth low energy*. Sebastopol, CA: O'Reilly Media, 2014.
[42] R. Heydon, *Bluetooth low energy: The developer's handbook*. Upper Saddle River, N.J.: Prentice Hall, 2013.
[43] Patrick O'Keeffe, *mqtt/mqtt.github.io*. [Online] Available: https://github.com/mqtt/mqtt.github.io/wiki/history. Accessed on: May 17 2017.
[44] *MQTT Essentials Part 3: Client, Broker and Connection Establishment*. [Online] Available: http://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment. Accessed on: May 19 2017.
[45] *MQTT Essentials Part 2: Publish & Subscribe*. [Online] Available: http://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe. Accessed on: May 19 2017.
[46] W.-J. Chen, *Responsive mobile user experience using MQTT and IBM MessageSight*, 1st ed. Poughkeepsie, NY: IMB Corp. International Technical Support Organization, 2014.
[47] Wikipedia, *MQTT - Wikipedia*. [Online] Available: https://en.wikipedia.org/w/index.php?oldid=780647016. Accessed on: May 19 2017.
[48] *HTTP: Brief History of HTTP - High Performance Browser Networking (O'Reilly)*. [Online] Available: https://hpbn.co/brief-history-of-http/. Accessed on: Jul. 12 2017.
[49] *HTTP request methods*. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods. Accessed on: Jul. 12 2017.
[50] L. Richardson and S. Ruby, *Web Services mit REST*. Beijing: O'Reilly, 2007.
[51] *Web API Endpoint Reference - Spotify Developer*. [Online] Available: https://developer.spotify.com/web-api/endpoint-reference/. Accessed on: Jul. 12 2017.
[52] P. Schoutsen, *Home Assistant – home automation in Python*. [Online] Available: http://paulusschoutsen.nl/blog/2013/12/home-assistant-home-automation-in-python/. Accessed on: Jun. 26 2017.
[53] P. Schoutsen, *Website launched!* [Online] Available: https://home-assistant.io/blog/2014/12/18/website-launched/. Accessed on: Jun. 26 2017.
[54] H. Assistant, *Components*. [Online] Available: https://home-assistant.io/components/#all. Accessed on: Jun. 26 2017.
[55] *home-assistant/home-assistant*. [Online] Available: https://github.com/home-assistant/home-assistant. Accessed on: Jun. 26 2017.
[56] H. Assistant, *Home Assistant*. [Online] Available: https://home-assistant.io/. Accessed on: Jul. 26 2017.
[57] H. Assistant, *Frontend of Home Assistant*. [Online] Available: https://home-assistant.io/docs/frontend/. Accessed on: Jun. 27 2017.
[58] H. Assistant, *iOS*. [Online] Available: https://home-assistant.io/docs/ecosystem/ios/. Accessed on: Jun. 27 2017.
[59] *Build software better, together*. [Online] Available: https://github.com/search?utf8=%E2%9C%93&q=home-assistant&type=Repositories. Accessed on: Jul. 10 2017.

[60] *home assistant - Google Search.* [Online] Available: https://www.google.de/search?q=home+assistant. Accessed on: Jul. 10 2017.
[61] *"home assistant" - Google Search.* [Online] Available: https://www.google.de/search?q=home+assistant#q=%22home+assistant%22. Accessed on: Jul. 10 2017.
[62] *"home assistant" -google -nursing -nurse - Google Search.* [Online] Available: https://www.google.de/search?q=home+assistant#q=%22home+assistant%22+-google+-nursing+-nurse. Accessed on: Jul. 10 2017.
[63] *Home Assistant Community.* [Online] Available: https://community.home-assistant.io/about. Accessed on: Jul. 25 2017.
[64] D. Higgs, *Home Assistant is moving to Discord.* [Online] Available: https://home-assistant.io/blog/2017/07/03/home-assistant-is-moving-to-discord/. Accessed on: Jul. 10 2017.
[65] *Rudolf König im Interview ' Der Erfinder von FHEM zum Thema Smart Home | meintechblog.de.* [Online] Available: http://www.meintechblog.de/2015/07/rudolf-koenig-im-interview-der-erfinder-von-fhem-zum-thema-smart-home/. Accessed on: Jul. 25 2017.
[66] *FHEM APP – FHEMWiki.* [Online] Available: https://wiki.fhem.de/wiki/FHEM_APP. Accessed on: Jul. 25 2017.
[67] *AndFHEM – FHEMWiki.* [Online] Available: https://wiki.fhem.de/wiki/AndFHEM. Accessed on: Jul. 25 2017.
[68] *Build software better, together.* [Online] Available: https://github.com/search?utf8=%E2%9C%93&q=fhem&type=. Accessed on: Jul. 25 2017.
[69] *fhem - Google Search.* [Online] Available: https://www.google.de/search?q=fhem&oq=fhem&aqs=chrome.69i57j69i60l5.193j0j4&sourceid=chrome&ie=UTF-8. Accessed on: Jul. 25 2017.
[70] *FHEM Forum - Index.* [Online] Available: https://forum.fhem.de/. Accessed on: Jul. 25 2017.
[71] *iOS openHAB App - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/addons/uis/apps/ios.html. Accessed on: Jul. 25 2017.
[72] *Android openHAB App - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/addons/uis/apps/android.html. Accessed on: Jul. 25 2017.
[73] *Build software better, together.* [Online] Available: https://github.com/search?utf8=%E2%9C%93&q=openhab&type=. Accessed on: Jul. 25 2017.
[74] *openhab - Google Search.* [Online] Available: https://www.google.de/search?q=openhab&oq=openhab&aqs=chrome.69i57j69i60l4j69i65.1468j0j4&sourceid=chrome&ie=UTF-8. Accessed on: Jul. 25 2017.
[75] *openHAB Community.* [Online] Available: https://community.openhab.org/about. Accessed on: Jul. 25 2017.
[76] T. S. House, *Aeotec by Aeon Labs Z-Stick DSA02203-ZWUS S2 USB Dongle.* [Online] Available: https://www.thesmartesthouse.com/products/aeotec-by-aeon-labs-z-stick-dsa02203-zwus-s2-usb-dongle. Accessed on: Jun. 24 2017.
[77] *USB-Controller Z-Stick S2 Z-Wave von Aeon Labs.* [Online] Available: https://www.generationrobots.com/de/401941-controller-z-stick-s2-z-wave-von-aeon-labs-.html. Accessed on: Jun. 24 2017.
[78] *Z-Wave Plus Aeotec MultiSensor | Vesternet.* [Online] Available: http://www.vesternet.com/z-wave-aeon-labs-multisensor-gen5. Accessed on: Jun. 24 2017.
[79] *Multisensor Gen5 user guide. : Aeotec by Aeon Labs.* [Online] Available: https://aeotec.freshdesk.com/support/solutions/articles/6000056451-multisensor-gen5-user-guide-. Accessed on: Jun. 24 2017.
[80] *Multisensor Gen5 technical specification. : Aeotec by Aeon Labs.* [Online] Available: https://aeotec.freshdesk.com/support/solutions/articles/6000167331-multisensor-gen5-technical-specification-. Accessed on: Jun. 24 2017.
[81] *Happy Bubbles Technology - How It Works.* [Online] Available: https://www.happybubbles.tech/presence/detector. Accessed on: Jun. 24 2017.
[82] *Home Automation using BLE Beacons and Happy Bubbles.* [Online] Available: https://www.youtube.com/watch?v=oBkahrDfUFE&t=4s. Accessed on: Jun. 24 2017.
[83] *Happy Bubbles Technology - How It Works.* [Online] Available: https://www.happybubbles.tech/presence/. Accessed on: Jun. 24 2017.
[84] *Happy Bubbles Technology - How It Works.* [Online] Available: https://www.happybubbles.tech/presence/beacon. Accessed on: Jun. 24 2017.
[85] *HomeMatic Zentrale CCU2 - eQ-3.* [Online] Available: http://www.eq-3.de/produkte/homematic/zentralen-und-gateways/homematic-zentrale-ccu-2.html. Accessed on: Jun. 24 2017.
[86] E. E. AG, *Homematic 103584 Zentrale CCU2 für Smart Home / Hausautomation | ELV-Elektronik.* [Online] Available: https://www.elv.de/homematic-zentrale-ccu-2.html. Accessed on: Jun. 26 2017.

[87] *For Developer - eQ-3.* [Online] Available: http://www.eq-3.com/topics/for-developer.html. Accessed on: Jun. 26 2017.
[88] *Homematic: CCU2 auf einen Raspberry Pi auslagern – RaspberryMatic › technikkram.net.* [Online] Available: https://www.technikkram.net/2016/12/homematic-ccu2-auf-einen-raspberry-pi-auslagern-raspberrymatic. Accessed on: Jun. 26 2017.
[89] *HomeMatic Funk-Dimmaktor 1-fach, Phasenabschnitt, Unterputzmontage - eQ-3.* [Online] Available: http://www.eq-3.de/produkte/homematic/licht/hm-lc-dim1t-fm.html#bestell_info. Accessed on: Jun. 26 2017.
[90] *HomeMatic Funk-Rollladenaktor 1-fach, Unterputzmontage - eQ-3.* [Online] Available: http://www.eq-3.de/produkte/homematic/rolllaeden-und-markisen/hm-lc-bl1-fm.html. Accessed on: Jun. 26 2017.
[91] S. Riedel, *SSH - kurz & gut,* 2nd ed. Köln: O'Reilly, 2006.
[92] G. L. D. Larisch, *TCP/IP? Grundlagen und Praxis, 2nd Edition*. [Place of publication not identified]: Dpunkt, 2013.
[93] *SSH-Verbindung via Terminal unter Mac OS X.* [Online] Available: http://macs-moritz.com/ssh-verbindung-via-terminal-unter-mac-os-x. Accessed on: Jun. 28 2017.
[94] K. Hemphill, *Introduction to Mac Terminal's tricks and shortcuts.* [Online] Available: http://www.macworld.co.uk/feature/mac-software/how-use-terminal-on-mac-3608274/. Accessed on: Jun. 28 2017.
[95] *Download PuTTY: latest release (0.70).* [Online] Available: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. Accessed on: Jul. 11 2017.
[96] *Documentation | Mosquitto.* [Online] Available: https://mosquitto.org/documentation/. Accessed on: Jul. 24 2017.
[97] H. Assistant, *Raspberry Pi All-In-One Installer.* [Online] Available: https://home-assistant.io/docs/installation/raspberry-pi-all-in-one/. Accessed on: Jun. 26 2017.
[98] H. Assistant, *Installation in virtualenv.* [Online] Available: https://home-assistant.io/docs/installation/virtualenv/. Accessed on: Jul. 11 2017.
[99] H. Assistant, *Hassbian.* [Online] Available: https://home-assistant.io/docs/hassbian/. Accessed on: Jun. 26 2017.
[100] H. Assistant, *Hass.io.* [Online] Available: https://home-assistant.io/hassio/. Accessed on: Jul. 26 2017.
[101] H. Assistant, *Zone.* [Online] Available: https://home-assistant.io/components/zone/. Accessed on: Jul. 19 2017.
[102] H. Assistant, *Automating Home Assistant.* [Online] Available: https://home-assistant.io/docs/automation/. Accessed on: Jul. 04 2017.
[103] H. Assistant, *MQTT Room Presence.* [Online] Available: https://home-assistant.io/components/sensor.mqtt_room/. Accessed on: Jul. 04 2017.
[104] H. Assistant, *Automation Editor.* [Online] Available: https://home-assistant.io/docs/automation/editor/. Accessed on: Jul. 24 2017.
[105] Paulus Schoutsen & Fabian Affolter, *Home Assistant 0.45: Automation editor, Z-Wave panel, OCR.* [Online] Available: https://home-assistant.io/blog/2017/05/20/automation-editor-zwave-panel-ocr/. Accessed on: Jul. 24 2017.
[106] *FHEM for Debian.* [Online] Available: https://debian.fhem.de/. Accessed on: Jul. 17 2017.
[107] *Erste Schritte in FHEM – FHEMWiki.* [Online] Available: https://wiki.fhem.de/wiki/Erste_Schritte_in_FHEM. Accessed on: Jul. 17 2017.
[108] *notify – FHEMWiki.* [Online] Available: https://wiki.fhem.de/wiki/Notify. Accessed on: Jul. 18 2017.
[109] *HomeMatic – FHEMWiki.* [Online] Available: https://wiki.fhem.de/wiki/HomeMatic#Struktur_von_HM_Ger.C3.A4ten. Accessed on: Jul. 18 2017.
[110] *HMCCU – FHEMWiki.* [Online] Available: https://wiki.fhem.de/wiki/HMCCU. Accessed on: Jul. 18 2017.
[111] *MQTT JSON auslesen.* [Online] Available: https://forum.fhem.de/index.php/topic,74253.msg659618.html#msg659618. Accessed on: Jul. 18 2017.
[112] *openHAB 2 on Linux - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/installation/linux.html. Accessed on: Jul. 09 2017.
[113] *Installation Overview - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/installation/index.html#prerequisites. Accessed on: Jul. 09 2017.
[114] *openHAB 2 has arrived - JAXenter.* [Online] Available: https://jaxenter.com/openhab-2-arrived-131328.html. Accessed on: Jul. 09 2017.
[115] *openHABian - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/installation/openhabian.html. Accessed on: Jul. 09 2017.
[116] *User Interfaces - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/addons/uis.html. Accessed on: Jul. 24 2017.
[117] *Configuration of openHAB.* [Online] Available: http://docs.openhab.org/tutorials/beginner/configuration.html. Accessed on: Jul. 24 2017.
[118] *Bindings - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/addons/bindings.html. Accessed on: Jul. 24 2017.

[119] *Things - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/concepts/things.html. Accessed on: Jul. 24 2017.
[120] *Items - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/concepts/items.html. Accessed on: Jul. 24 2017.
[121] *Creating a sitemap - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/tutorials/beginner/sitemap.html. Accessed on: Jul. 24 2017.
[122] *Sitemaps - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/configuration/sitemaps.html. Accessed on: Jul. 24 2017.
[123] *Logging - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/administration/logging.html#karaf-console. Accessed on: Jul. 24 2017.
[124] *Rules - openHAB 2 - Empowering the Smart Home.* [Online] Available: http://docs.openhab.org/configuration/rules-dsl.html. Accessed on: Jul. 24 2017.
[125] S. Efftinge and M. Spoenemann, *Xtend - Modernized Java.* [Online] Available: http://www.eclipse.org/xtend/index.html. Accessed on: Jul. 24 2017.

Eidesstattliche Versicherung zur Bachelorarbeit

Ich versichere, die Bachelorarbeit selbstständig und lediglich unter Benutzung der angegebenen Quellen und Hilfsmittel verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß veröffentlichtem oder nicht veröffentlichtem Schrifttum entnommen sind, habe ich als solche kenntlich gemacht.

Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Offenburg, den 27. Juli 2017

_____